CHALMERS
UNIVERSITY OF TECHNOLOGY

Traction Control of Heavy Vehicles using Road Information

A Nonlinear Model Predictive Control framework that incorporates future road information to optimize wheel-slip and maximize traction.

Master's thesis in Systems, Control and Mechatronics

GEORGE VASANTH FRANSUA SIMON
VALDEMAR SAMUELSSON

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master thesis 2025

Traction Control of Heavy Vehicles using Road Information

A Nonlinear Model Predictive Control framework that incorporates future road information to optimize wheel-slip and maximize traction.

GEORGE VASANTH FRANSUA SIMON
VALDEMAR SAMUELSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2025

Traction Control of Heavy Vehicles using Road Information.
A Nonlinear Model Predictive Control framework that incorporates future road information to optimize wheel-slip and maximize traction.
GEORGE VASANTH FRANSUA SIMON
VALDEMAR SAMUELSSON

Supervisors:
Basar Özkan, Volvo Group Trucks Technology
Esteban Gelso, Volvo Group Trucks Technology

Examiner:
Nikolce Murgovski, Electrical Engineering

Master Thesis 2025
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A Volvo FMX truck powers through a muddy road, where its wheels are slipping, which is an example of traction loss.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2025

Traction Control of Heavy Vehicles using Road Information.
A Nonlinear Model Predictive Control framework that incorporates
future road information to optimize wheel-slip and maximize traction.
GEORGE VASANTH FRANSUA SIMON
VALDEMAR SAMUELSSON
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Loss of traction, especially while climbing a hill on low-friction surfaces with heavy vehicles, significantly compromises safety and performance. This thesis presents an approach that integrates future road information, such as friction coefficients and slope, to control vehicle velocity and optimize wheel slip in real time. Using road information data, the controller setup anticipates the wheel speed needed for a reference velocity and gives the necessary torque to be at that wheel speed while also adhering to the constraints. A decoupled architecture with three modules is used to leverage different control systems for different subproblems. Firstly, a differential lock controller decides which differential setting should be used. Secondly, a Model Predictive Controller (MPC) tracks a reference velocity while adhering to constraints. Finally, a Wheel Speed Controller (WSC) is used to give out the necessary torque to control the wheel speed. The decoupling is mainly done to keep the computation times close to real-time implementations for heavy-duty truck platforms. Extensive simulations were done on both synthetic and real-life data to validate the controller pipeline for varying road profiles. Overall, the results indicate that preview based traction control can noticeably improve operational safety and performance for heavy vehicles in off-road and low-friction scenarios.

Keywords: Model Predictive Control, Differentials, Trucks, Traction Control, low-friction scenarios, road grade.

Acknowledgements

We extend our heartfelt gratitude to our thesis supervisors at Volvo Group Trucks Technology, Basar Özkan and Esteban Gelso, for their unwavering support, insightful guidance, and trust throughout this project. Your encouragement and constructive feedback were instrumental in shaping the thesis into its final form.

We are also deeply grateful to our examiner, Professor Nikolce Murgovski, for his continued support and valuable expertise. Your thoughtful critiques and guidance have greatly enriched our work.

Our thanks go to all members of the Performance and Prediction team within the Vehicle Motion Management group at Volvo Group Trucks Technology. Thank you for creating such a welcoming environment and for generously sharing your knowledge and experience. We are likewise appreciative of our fellow thesis colleagues in the VMM team, all the engaging discussions and moments of laughter have made this journey all the more rewarding.

Finally, we wish to acknowledge our families and friends for their continuous encouragement and thoughtful feedback. Your support has sustained us through every step of this journey.

George Vasanth Fransua Simon, Gothenburg, May 2025
Valdemar Samuelsson, Gothenburg, May 2025

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
VTM	Volvo Transport Model
OCP	Optimal Control Problem
NLP	Non Linear Programming
IPOPT	Interior Point OPTimizer
FATROP	FAst TRajectory OPtimizer
BONMIN	Basic Open source Nonlinear Mixed INteger programming
WSC	Wheel Speed Controller

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Indices

i	Index for step in MPC horizon
j	Index for axle number, increasing from front to rear
t	Index for time step

Sets

$\mathbb{X}$	Set of state variables
$\mathbb{U}$	Set of control variables
$\mathbf{w}$	Set of decision variables
$\mathbf{X}$	Boolean set representing the status of differential lock setting
$\mathcal{X}$	Admissible set of state variables
$\mathcal{U}$	Admissible set of control variables

Parameters

F_{zj}	Normal force on axle i
r	Radius of tire
θ	Road incline angle
m	Mass of the vehicle
g	Gravity constant
F_{RR}	Forces due to rolling resistance of wheels
F_{Drag}	Aerodynamic drag

I_j	Rotational inertia of axle j
B	Stiffness factor of the tire-road interaction
C	Shape factor of the tire-road interaction
D	Peak force of the tire-road interaction
E	Curvature factor of the tire-road interaction
N	Prediction horizon length of MPC

Variables

v	Vehicle velocity
$v_{\max}$	Maximum allowed velocity
$v_{\min}$	Minimum allowed velocity
v_{ref}	Reference velocity
a	Vehicle acceleration
ω_j	Angular velocity of axle j
X_1	Front axle differential lock activation status
X_2	Rear axle differential lock activation status
κ	Slip ratio of axle
τ	Torque from the engine
T_s	Sampling time
F_{xj}	Forward traction force from axle j

Contents

List of Figures

List of Tables

1
Introduction

1.1 Background

Traction control systems have become indispensable in modern vehicles by preventing wheel slip and optimizing tire-road interaction to enhance both safety and performance [1]. In recent decades, they have helped reduce road fatalities by assisting drivers on low-grip surfaces and by explicitly accounting for actuator limits and constraints during the control design process, ensuring robust operation even under the harshest conditions.

Commercial vehicles operate in diverse conditions, including wet, icy, and uneven terrain, where maintaining adequate traction is critical both for efficiency and accident prevention. Current traction control systems rely on real-time vehicle data to minimize wheel slip, but typically react only to immediate road conditions and do not anticipate upcoming terrain variations such as incline or road surface changes. This limitation can cause a truck to ascend a hill or slope too slowly, risking loss of momentum or even sliding backward on steep grades.

Traditional approaches to traction and stability control often employ PID [4] or sliding mode controllers [5], and while the Model Predictive Control (MPC) approach has been adopted in various vehicle stability contexts [6], it remains underutilized in dedicated traction systems. There are papers using model predictive control for traction control [7] [8], but they focus on only controlling wheel slip at the current time without looking into future road information. Both use individually controlled axles driven by electric motors, where one controller is used for each wheel/axle. This is different compared to diesel trucks where there is only one engine used, where there is one control input for all wheels.

Current research has shown that incorporating future surface friction data [2] [3] significantly enhances traction control performance. However, relatively little work has been done in exploring the simultaneous inclusion of future road-grade information as well. By leveraging both anticipated friction and road grade, one can accurately compute the minimum speed and traction needed to tackle demanding segments such as steep inclines or slippery terrain ahead of time. Crucially, this predictive strategy optimizes traction along the entire route, ensuring consistent grip and stability rather than merely reacting to immediate conditions.

1.2 Research Questions

This thesis aims to address the following research questions:

1. What performance improvements can be achieved by using a Model Predictive Control (MPC) based traction controller that utilizes future road information?
2. How does the minimum safe speed for hill ascent vary with road grade, surface friction, and vehicle dynamics, and can this speed be computed in real time to prevent loss of momentum or rollback?
3. Under what driving conditions should differential locks be automatically engaged or disengaged to maintain optimal traction while minimizing unnecessary wear and energy consumption?
4. Finally, is it possible to integrate all the above components into a single unified system?

1.3 Scope and Limitations

The focus of this thesis is on the longitudinal traction behavior under predictive control. Specifically:

1. **Longitudinal dynamics only:** Lateral and Vertical vehicle dynamics are neglected.
2. **Simulation validation:** The controller is validated in MATLAB/Simulink's Volvo Transport Model (VTM), not through field trials on a physical truck.
3. **No braking control:** Braking actions are excluded from the controller design.
4. **Vehicle baseline:** All development and simulations are based on a Volvo FMX 6×6 model.

1.4 Ethical and Sustainability Aspects

Working on traction control for heavy vehicles is not just a technical exercise. It is about making roads safer. These systems help vehicles stay stable on slippery surfaces, cutting down accidents and keeping people protected. They also advance the United Nations' Sustainable Development Goals, especially SDG 3 [10], SDG 9 [11], and SDG 11 [12] preventing crashes and smoothing traffic flow.

Since the proposed traction control algorithm is developed with the possibility of being able to run on existing electronic control units, it does not need new hardware. That means extra environmental impacts from manufacturing can be avoided. Instead, the focus was on developing an efficient way that minimizes energy draw and on choosing a way that does not necessarily require a huge upgrade on existing hardware.

Enabling heavy trucks to climb steep inclines more effectively has broader implications for logistics efficiency and fuel consumption. Poor traction on hills often leads to wheel slip, excessive fuel use, and longer travel times due to reduced speeds or stalled vehicles. By improving the capabilities to climb hills, the system can help

reduce these inefficiencies. This contributes to more sustainable transport and supports the shift towards greener supply chains. Additionally, reducing the likelihood of vehicles getting stuck on inclines helps avoid road blockages and the associated disruptions, which is beneficial for both economic productivity and public infrastructure usage.

2
Theory

2.1 Vehicle Coordinate System

A vehicle-fixed coordinate system (X, Y, Z) is defined as follows:

- X (longitudinal axis): Points forward along the vehicle's centerline, in the direction of travel.
- Y (lateral axis): Points to the right of the vehicle, perpendicular to the longitudinal axis, in the horizontal plane.
- Z (vertical axis): Points downward, perpendicular to both the longitudinal and lateral axes.

All translational velocities and accelerations can be expressed as components $\{v_x,\, v_y,\, v_z\}$ along $\{X,\, Y,\, Z\}$, respectively, and forces as $\{F_x,\, F_y,\, F_z\}$.

2.2 Longitudinal Vehicle Dynamics

The longitudinal dynamics of the vehicle are based on the sum of all longitudinal forces act along its direction of motion. These forces acts longitudinally on a vehicle moving on an inclined road, as shown in Figure 2.1. Resistive forces that oppose the vehicle's motion include aerodynamic drag, rolling resistance, and the component of gravitational force parallel to the road incline but opposite to direction of travel. The primary driving forces contributing to acceleration are the traction forces generated from all driven tires. By applying Newton's Second Law along the longitudinal axis, these forces can be combined to yield the equation (2.1) which describes how the vehicle velocity changes based on the forces acting on it.

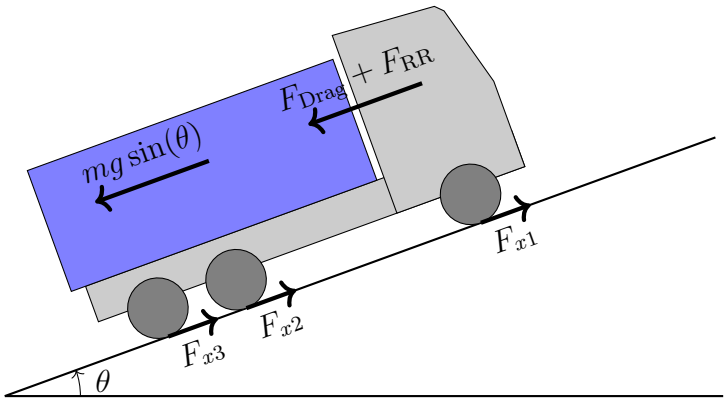


Figure 2.1: Figure of the forces active in the longitudinal vehicle dynamics

$$m\dot{v} = m\frac{dv}{dt} = F_{x1} + F_{x2} + F_{x3} - mg\sin(\theta) - F_{\text{Drag}} - F_{\text{RR}} \tag{2.1}$$

2.3 Magic Formula Tire Model

The traction force generated by the tires can be estimated using the Pacejka tire model, commonly referred to as the "Magic Formula". Developed by Hans B. Pacejka [18], this empirical model characterizes tire behavior under various operating conditions by defining the relationship between tire forces and slip parameters through a set of fitted equations. As such, it is extensively used in vehicle dynamics simulations and control systems.

The general form for the magic formula without accounting for longitudinal or horizontal offset is shown in equation (2.2) with slip defined in equation (2.3). This formulation yields a characteristic force-slip curve, as illustrated in Figure 2.2, which depends on specific tire parameters.

While the model is based on curve fitting rather than first-principles physics, it effectively captures longitudinal tire behavior. However, it remains a simplification and does not fully represent the complete dynamics of tire-road interaction.

$$F_x = F_z D sin[C\arctan\{B\kappa - E(B\kappa - \arctan(B\kappa))\}] \tag{2.2}$$

$$\kappa = \frac{\omega r - v}{\omega r} \tag{2.3}$$

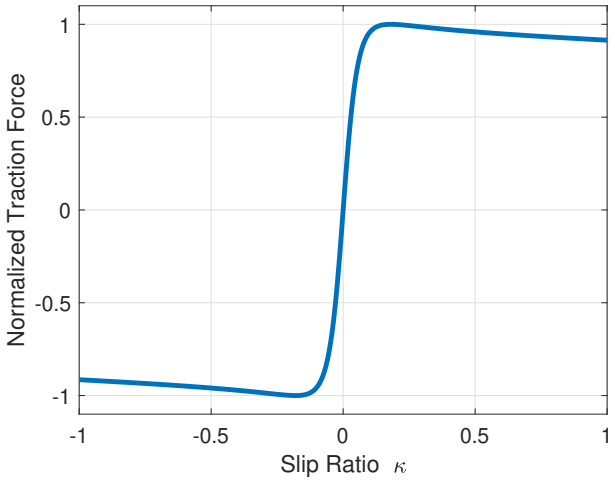


Figure 2.2: Example of the shape of the "magic formula"

2.4 Wheel Dynamics

The angular velocity of the wheel, which is used to calculate slip, is influenced by its angular acceleration. This acceleration results from the applied torque, the tractive force from the tire-road interaction, and the wheel's moment of inertia. Combining these dynamics yields the equation (2.4) which describes the wheel's angular acceleration based on the net torque acting on it.

$$\dot{\omega} = \frac{d\omega}{dt} = \frac{\tau_j - F_{xj} r}{I_j} \tag{2.4}$$

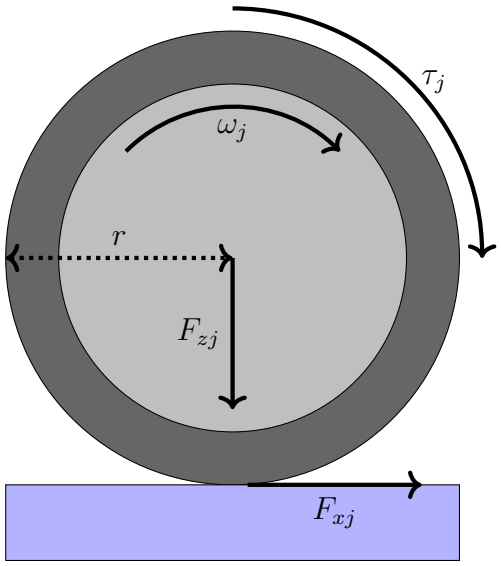


Figure 2.3: Figure showing the forces acting on a wheel

2.5 Differentials

The torque generated by the engine is transmitted through a series of differentials before reaching the wheels. The truck is equipped with multiple differentials, each of which can operate in either an open or locked configuration. In an open differential, the input torque τ_0 is distributed equally between the output shafts τ_1, τ_2 as shown in equation (2.5). Additionally, the average angular velocity of the output shafts ω_1 and ω_2 equals the angular velocity of the input shaft ω_0, as shown in equation (2.6).

$$\frac{\tau_0}{2} = \tau_1 = \tau_2 \tag{2.5}$$

$$\omega_0 = \frac{\omega_1 + \omega_2}{2} \tag{2.6}$$

In contrast, a locked differential constrains the output shafts to spin at the same angular velocity as the input shaft, shown in equation (2.7). In this case the distribution of output torques is not defined, but the sum of the output torques remains equal to the input torque as in equation (2.8).

$$\omega_0 = \omega_1 = \omega_2 \tag{2.7}$$

$$\tau_0 = \tau_1 + \tau_2 \tag{2.8}$$

While locking the differential enhances traction, particularly in low-traction environments, it also introduces certain trade-offs. Specifically, it can lead to increased fuel consumption, accelerated tire wear, and reduced steering performance. Therefore, differential locking should be employed sparingly and only when necessary to optimize vehicle performance.

2.6 Differential Layout

The Volvo FMX 6x6 truck that has been taken into consideration is equipped with five differentials arranged in the configuration illustrated in Figure 2.4. Their functions are described below:

1. **Front inter-axle lock:** This connects the front axle to the rear axles. When disengaged, all engine torque is directed exclusively to the rear inter-axle differential, with no torque transmitted to the front axle. When engaged, it functions as a locked differential.
2. **Rear inter-axle differential:** This differential connects the two rear axles to the front inter-axle lock.
3. **Front axle inter-wheel differential:** This differential connects the two front wheels to the front inter-axle lock.
4. **Middle axle inter-wheel differential:** This differential connects the two middle wheels to the rear inter-axle differential.
5. **Rear axle inter-wheel differential:** This differential connects the two rear wheels to the rear inter-axle differential.

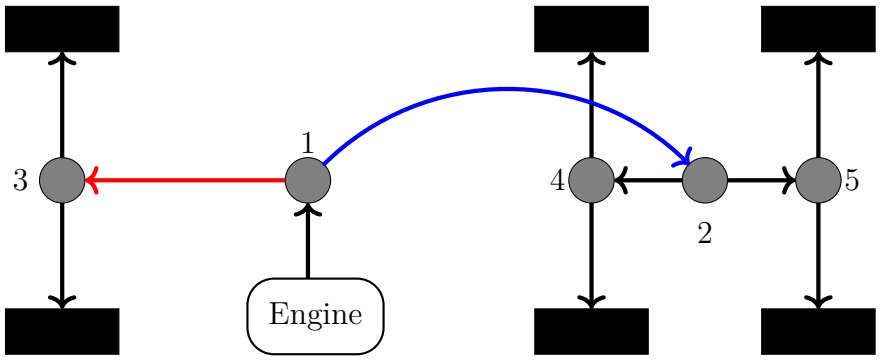


Figure 2.4: Diagram of the differential system. Arrows indicate the inputs and outputs to each differential.

These differentials are combined into 5 different available settings:

1. **Configuration 1**: No differential locks active.
2. **Configuration 2**: Rear inter-axle lock active (number 2 in the image).
3. **Configuration 3**: Front and rear inter-axle locks active (numbers 1 and 2 in the image).
4. **Configuration 4**: Rear inter-wheel locks and both inter-axle locks active (numbers 1,2,4 and 5 in the image).
5. **Configuration 5**: All differential locks locked. (numbers 1,2,3,4 and 5 in the image)

In the current controller setup only Configurations 1, 2, and 3 are actively managed.

2.7 Volvo Transport Model

Volvo Transport Model (VTM) [13] is a detailed Simulink based vehicle dynamics simulator developed by Volvo GTT. It produces accurate responses to driver inputs like wheel torques, steering angle, and road conditions by modeling the full truck and trailer dynamics. For differential lock behavior, a separate Simulink model (also provided by Volvo GTT) is employed. It accepts a single input torque and dynamically distributes it between the wheels based on the difflock setting, ensuring realistic torque distribution and handling under varying loads and surface conditions.

2.8 Model Predictive Control

Model Predictive Control (MPC) is an advanced control strategy widely used in vehicle dynamics, robotics, and industrial automation. MPC utilizes a mathematical model of the system to predict its future states over a predefined time horizon. At each time step, an optimization problem is solved to determine the optimal sequence of control inputs that minimize a cost function while respecting system constraints. This predictive capability enables MPC to handle multi-variable systems with constraints more effectively than traditional control methods like PID controllers.

The fundamental working principle of MPC involves iteratively solving an optimization problem that balances system performance and constraint satisfaction. Once the optimal control sequence is determined, only the first control input is applied, and the process is repeated at the next time step with updated system measurements. MPC has computational challenges, as solving the optimization problem in real-time requires significant computational power.

2.8.1 Mathematical Formulation of MPC

In MPC, at each sampling instant, a finite-horizon Optimal Control Problem (OCP) is solved. Given the current state $\mathbf{x}_0$, it finds the control sequence $\{\mathbf{u}_0, \ldots, \mathbf{u}_{N-1}\}$ that minimizes the cumulative cost over N steps, while respecting the system dynamics and admissible sets.

$$\begin{aligned}
\min_{\mathbf{u}} \quad & J_N(\mathbf{x}_0, \mathbf{u}) = \sum_{t=0}^{N-1} \ell(\mathbf{x}_t, \mathbf{u}_t) && \text{(2.9a)}\\
\text{subject to} \quad & \mathbf{x}_u(t+1) = \mathbf{f}(\mathbf{x}_u(t), \mathbf{u}(t)), && \text{(2.9b)}\\
& \mathbf{x}_u(0) = \mathbf{x}_0, && \text{(2.9c)}\\
& \mathbf{u}_t \in \mathcal{U}, \quad \forall t \in [0, N-1], && \text{(2.9d)}\\
& \mathbf{x}_t \in \mathcal{X}, \quad \forall t \in [0, N]. && \text{(2.9e)}
\end{aligned}$$

2.8.1.1 Nonlinear Programming Problem Formulation

To solve an optimal control problem (OCP) with off-the-shelf nonlinear programming (NLP) solvers, it must first be transcribed into a standard NLP form:

$$\begin{aligned}
\min_{\mathbf{w}} \quad & \phi(\mathbf{w}), && \text{(2.10a)}\\
\text{s.t.} \quad & g_1(\mathbf{w}) \leq 0, && \text{(2.10b)}\\
& g_2(\mathbf{w}) = 0. && \text{(2.10c)}
\end{aligned}$$

Here, $\mathbf{w}$ collects all decision variables, $\phi(\mathbf{w})$ is the cost function, $g_1(\mathbf{w})$ encodes any inequality constraints (e.g., state or input bounds), and $g_2(\mathbf{w})$ encodes equality constraints (e.g., system dynamics and boundary conditions).

There are two common ways to turn an OCP into this form: *single shooting* and *multiple shooting*. Each approach handles the system dynamics differently when building the decision vector and constraining the problem.

1. Single Shooting. In single shooting, the only decision variables are the control inputs over the horizon which is shown in equation (2.11)

$$\mathbb{U} = \begin{bmatrix} u_0 & u_1 & \cdots & u_{N-1} \end{bmatrix} = \mathbf{w}. \tag{2.11}$$

The state trajectory is not an independent decision variable; instead, it is generated by forward simulating the discretized dynamics from the known initial state x_0. If the continuous-time dynamics are discretized as

$$x_{t+1} = f\big(x_t, u_t\big) \tag{2.12}$$

then, for a given sequence $\{u_0, \ldots, u_{N-1}\}$ the following

$$x_1 = f(x_0, u_0), \quad x_2 = f\Big(x_1, u_1\Big), \ \ldots, \ x_N = f\Big(x_{N-1}, u_{N-1}\Big). \tag{2.13}$$

are computed. Since these states depend implicitly on the controls, they must be enforced as equality constraints in the NLP. Concretely let

$$\mathbb{X} = \begin{bmatrix} x_0 & x_1 & \ldots & x_N \end{bmatrix} \tag{2.14}$$

then the continuity constraints become

$$g_2(\mathbf{w}) \ = \ \begin{bmatrix} x_1 - f(x_0, u_0) \\ x_2 - f(x_1, u_1) \\ \vdots \\ x_N - f(x_{N-1}, u_{N-1}) \end{bmatrix} = 0. \tag{2.15}$$

Notice that x_0 is known initial condition, so the first line $x_1 - f(x_0, u_0) = 0$ effectively ties u_0 to x_1, and so on. In practice, one often omits the explicit appearance of $x_1, \ldots, x_N$ in the decision vector since they are computed during simulation.

From the solver's perspective, each time it chooses a candidate control sequence $\mathbf{w} = \mathbb{U}$, the states are forward-simulated internally, and these equations must hold exactly. While this single-shooting approach keeps the decision vector small (only N inputs), it becomes strongly nonlinear in $\mathbf{w}$, especially if the prediction horizon N is large or the dynamics $f(\cdot)$ are highly nonlinear. As a result, the solver may struggle with convergence, because a small change in an early control can drastically alter the end-of-horizon state.

2. Multiple Shooting. To reduce the strong nonlinear coupling between u_t and later states, all intermediate states can be *lifted* (i.e., explicitly included) as decision variables alongside the controls. In other words, the decision vector becomes (2.16)

$$\mathbf{w} \ = \ \Big\{ u_0, \ldots, u_{N-1}, \ x_1, \ldots, x_N \Big\}. \tag{2.16}$$

Here, x_t is treated as an independent optimization variable. Of course, these x_k must still satisfy the system dynamics. To enforce that, an alternate continuity constraint is introduced at each stage as opposed to how it was done with single shooting:

$$x_{t+1} \ - \ f\Big(x_t, \, u_t\Big) \ = \ 0, \quad t = 0, 1, \ldots, N-1. \tag{2.17}$$

Since x_0 is known (the initial condition), the first constraint enforces $x_1 = f(x_0, \, u_0)$. Then, independently for $t = 1$, the solver ensures $x_2 = f(x_1, \, u_1)$, and so on. In this way, each *shooting node* (x_t, u_t) is only locally coupled to (x_{t+1}, u_{t+1}) via a single constraint, rather than having all controls chain through a forward simulation as previously shown with the formation of $\mathbb{X}$ in equation (2.13). By including all x_t in $\mathbf{w}$, the NLP solver sees a sparser Jacobian structure and is numerically more robust, especially for long horizons or stiff dynamics.

In summary, both methods enforce exactly the same dynamic equations (2.12) and use the same cost $\phi(\cdot)$ and any path or terminal constraints. The difference lies in how the states are handled:

- **Single Shooting:**
  - Decision variables: $\{u_0, \ldots, u_{N-1}\}$ only.
  - States $x_1, \ldots, x_N$ are *implicitly* defined by forward simulation.
  - Constraints $g_2(\mathbf{w})$ enforce that the simulated states match the desired trajectory.
  - Pros: fewer optimization variables; straightforward to implement when N is small.
  - Cons: highly nonlinear dependence of x_N on early u_t can cause numerical difficulties.
- **Multiple Shooting:**
  - Decision variables: $\{u_0, \ldots, u_{N-1},\ x_0, \ldots, x_N\}$.
  - Each state x_t is an independent variable, and continuity $x_{t+1} = f(x_t, u_t)$ is enforced by (2.17).
  - Pros: sparser, better-conditioned Jacobian; improved solver convergence, especially for longer horizons.
  - Cons: more decision variables (since the state variables are included at each of N time steps), so the problem size is larger.

Thus, the choice between single and multiple shooting hinges on a trade-off between problem size and nonlinearity. For short horizons or mildly nonlinear systems, single shooting is often sufficient as it's more intuitive and easier to implement. For longer horizons, stiff or highly non-linear dynamics, or applications where solver robustness is paramount, multiple shooting is usually preferred.

2.8.2 OCP Frameworks

Experimentation with both CasADi [9] and Acados [15] in the pipeline revealed that each brings distinct strengths. With CasADi's symbolic framework (accessed via MATLAB), the states, inputs, transcription strategies, and cost functions can be modeled naturally. With its built in algorithmic differentiation, there tends to be fast and accurate evaluations of Jacobian and Hessians which are critical to solve a OCP problem. Switching between nonlinear programming solvers such as IPOPT, FATROP, or BONMIN requires minimal code modification thereby facilitating with convenient prototyping. The entire MPC problem can be defined symbolically, tested interactively, and cost weights or constraints tweaked on the fly without any C-code generation or recompilation.

Built on top of CasADi's symbolic expressions, Acados generates a dedicated C-code based implementation leveraging lightweight linear-algebra routines to solve the same OCP problem via block-structured Riccati recurrences with partial condensing and warm-starting to deliver predictable, low-latency performance. In theory, Acados solve times are usually much faster when compared to a generic CasADi–IPOPT setup on the target hardware, albeit at the expense of a slightly more involved configuration process (templating solver parameters, generating C code, and compiling).

As cited in [15] it can be seen that the Acados setup in comparison with the IPOPT setup is about *50*x faster on average. Overall, using CasADi for early-stage modeling and analysis and then switching to Acados for embedded execution enables rapid iteration in Python/MATLAB while achieving a high-performance, real-time solver without rewriting the core problem formulation.

2.8.2.1 OCP Solvers

IPOPT [16] is a versatile solver for general nonlinear problems that treats each optimization task as a "black box." It works by staying inside the feasible region and gradually moves toward a solution. To do this, IPOPT uses a strategy that balances making progress with avoiding poor or non-optimal points, essentially checking each step to ensure it leads closer to an optimum. Behind the scenes, it handles large, sparse systems of equations efficiently by relying on specialized sparse-matrix routines, and it automatically rescales variables and adds small adjustments to improve numerical stability. These features help IPOPT tackle a wide range of problems without needing manual tuning of every detail.

FATROP [17] on the other hand is designed specifically for time-ordered trajectory problems, such as those found in optimal control. Instead of solving one large system all at once, it breaks the problem into a sequence of smaller time-step problems and solves them in order. This often makes each solve much faster for structured, sequential tasks. FATROP also allows users to provide a good initial guess for the entire trajectory, which speeds up convergence when solving similar problems repeatedly. Finally, its internal convergence checks and update rules are tuned for the kinds of constraints and feasible regions that typically arise in trajectory optimization, rather than treating every problem as if it were arbitrary.

3
Problem Formulation

3.1 Objective Formulation

The primary control objective is to ensure accurate tracking of a reference velocity while maintaining the vehicle's speed within the predefined constraints specified in equation (3.4), across varying inclines and road conditions. In addition, the use of differential locks should be minimized. The associated cost function is defined in equation (3.1), where the control vector $\mathbf{X}$ represents the two actively controlled difflocks, as introduced in equation (3.2). The weighting matrices Q and R are diagonal matrices. Since the differential locks operate in a binary manner (either engaged or disengaged), their control inputs are modeled as Boolean variables, as shown in equation (3.3). The vehicle acceleration is a result of the longitudinal vehicle dynamics, where F_{xj} comes from the tire model in equation (3.7). The front inter-axle lock contributes to the dynamics only when it is engaged, thereby activating F_{x1}. Making F_{x1} controlled by the difflock controller changes the vehicle dynamics from equation (2.1) to equation (3.5). Due to the near equal rotational speeds of all driven wheels when the rear inter-axle locks are engaged, a unified slip ratio can be used across all wheels, as indicated in equation (3.8). Consequently, rather than modeling each axle independently using equation (2.4), all driven axles are aggregated into a single equivalent axle, leading to the simplified vehicle dynamics described in equation (3.6).

3.2 Problem Definition

Objective: Minimize the deviation from the reference velocity v_{ref} given a sequence of inclines θ and Pacejka tire-road parameters B, C, D, E:

$$J = \int_0^T ((v - v_{\text{ref}})^\top Q(v - v_{\text{ref}}) + \mathbf{X}^\top R\mathbf{X})dt \tag{3.1}$$

$$\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \tag{3.2}$$

Subject to:

$$\mathbf{X} \in \{0, 1\} \tag{3.3}$$

$$v_{\min} \leq v \leq v_{\max} \tag{3.4}$$

$$\dot{v} = \frac{dv}{dt} = \frac{F_{x1}X_1 + F_{x2} + F_{x3} - mg\sin(\theta) - F_{\text{Drag}} - F_{\text{RR}}}{m} \tag{3.5}$$

$$\dot{\omega} = \frac{d\omega}{dt} = \frac{\tau - F_{x1}rX_1 - F_{x2}r - F_{x3}r}{I_1X_1 + I_2 + I_3} \tag{3.6}$$

$$F_{xj} = f(\kappa, F_{zj}) = F_{zj}D\sin\Big(C\arctan\Big(B\kappa - E(B\kappa - \arctan(B\kappa))\Big)\Big) \tag{3.7}$$

$$\kappa = \frac{\omega r - v}{\omega r} \tag{3.8}$$

3.2.1 Open Rear Inter-axle Lock

For the case where the rear inter-axle lock is open, i.e., $X_2 = 0$, the two rear axles are no longer constrained to rotate at the same angular velocity. Equation (3.6) remains valid, assuming that ω represents the angular velocity of the fastest spinning axle. To maintain this assumption, the longitudinal forces F_{x2} and F_{x3} must be appropriately adjusted. Since the torques are equally distributed between the axles as in equation (2.5), at steady state equation (3.9) holds true.

$$F_{x2} = F_{x3} \tag{3.9}$$

The traction force is therefore limited by the wheel with the least traction. That is where the maximum F_{xj} is the lowest.

4

Controller Design

4.1 Controller Setup

The controller is divided into three parts (see Figure 4.1). The first part is a long-term MPC that follows a reference velocity and gives a reference slip as control input, which is converted to desired angular velocity. The second part is a wheel speed controller that tries to keep the desired angular velocity from the long-term MPC by giving a torque input. The third part is a difflock controller that activates the front and rear interaxle locks. It is divided between long-term MPC and wheel speed controller because the wheel speed controller needs to run at a high frequency, for which the computation time of the long-term MPC is too high. The nonlinearities of the wheel dynamics also give it a longer computation time and make the computation less stable. The difflock controller is separated from the MPC because it has discrete actions as either on or off, while the MPC optimizes over continuous values.

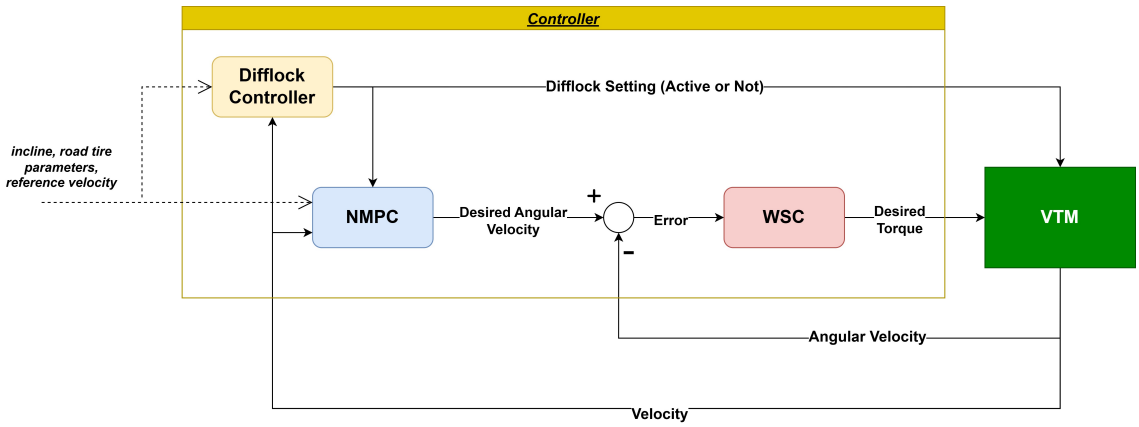


Figure 4.1: Block diagram of the controller

4.2 Modifying Road Parameters for the Controller

To model how road characteristics change over a distance step Δs, averaged or effective parameters are computed from discrete data available at specific intervals. These effective parameters are then what are used by the MPC and diff-lock controller.

4.2.1 Locked Rear Differential

For the cases where no open differentials are used, then all axles spin at the same angular velocity. This means that the force from each axle can be taken individually. This is used to get the average road surface that is then used by the long-term MPC and the difflock controller. To represent the road surface an axle experiences during each prediction distance, the average over this distance is used.
The road information is formatted as discrete data points at a certain sampling distance. In order to get the road parameters for a certain distance interval, the discrete parameters are interpolated into continuous parameters based on distance. The interpolated parameters are then $\theta(s), B(s), C(s), D(s)$ and $E(s)$.

For the incline θ the average incline $\bar{\theta}_{i,j}$ experienced by an axle over a prediction distance is arcsin of the change in height over that distance.

$$\bar{\theta}_{i,j} = \arcsin(\frac{h(s_{i,j}) - h(s_{i-1,j})}{s_{i,j} - s_{i-1,j}}) \tag{4.1}$$

For the road tire interaction, the average of the peak value D is found by the integral over the prediction distance divided by the length of the prediction distance (4.2).

$$\bar{D}_{i,j} = \frac{1}{\Delta s} \int_{s_{i-1}}^{s_i} D_j(s) ds \tag{4.2}$$

For the other road-tire parameters, just averaging them is not enough since to get the most traction, the tire should be closest to optimal slip at the highest traction point. To get a more accurate representation of the parameters, they are weighted by the peak value, as seen in equations (4.3), (4.4) and (4.5). This gives a higher importance to parameters at the points where the friction coefficient is higher.

$$\bar{B}_{i,j} = \frac{1}{\Delta s \int_{s_{i-1}}^{s_i} D_j(s) ds} \int_{s_{i-1}}^{s_i} B_j(s) D_j(s) ds \tag{4.3}$$

$$\bar{C}_{i,j} = \frac{1}{\Delta s \int_{s_{i-1}}^{s_i} D_j(s) ds} \int_{s_{i-1}}^{s_i} C_j(s) D_j(s) ds \tag{4.4}$$

$$\bar{E}_{i,j} = \frac{1}{\Delta s \int_{s_{i-1}}^{s_i} D_j(s) ds} \int_{s_{i-1}}^{s_i} E_j(s) D_j(s) ds \tag{4.5}$$

Based on the average parameter for each axle, the average between the axles is taken based on the normal force on the axle. This is done through equation (4.6) where $\bar{P}_{i,j}$ is defined in equation (4.6)

$$\bar{P}_i = \frac{f_{z1} \bar{P}_{i,1} X_1 + f_{z2} \bar{P}_{i,2} + f_{z3} \bar{P}_{i,3}}{f_{z1} X_1 + f_{z2} + f_{z3}} \tag{4.6}$$

$$\bar{P}_{i,j} = \{\bar{\theta}_{i,j}, \bar{B}_{i,j}, \bar{C}_{i,j}, \bar{D}_{i,j}, \bar{E}_{i,j}\} \tag{4.7}$$

4.2.2 Unlocked Rear Differential

For the case when the rear differential is open, each of the rear axles can spin at a different speed, but they still receive the same amount of torque. As a result, the total torque the system can use is limited to twice the torque of the axle with the least grip. This means the road parameters that should be used at every predicted point should be based on the axle with the least grip, see equation (4.8). Using the parameters from the axle with the least traction, the average is taken according to equation (4.10) where P_j is defined in equation (4.9).

$$\bar{P}(s) = \begin{cases} F_{z2}(s)D_2(s) \leq F_{z3}(s)D_3(s) \Rightarrow \bar{P}(s) = P_2(s) \\ F_{z2}(s)D_2(s) > F_{z3}(s)D_3(s) \Rightarrow \bar{P}(s) = P_3(s) \end{cases} \tag{4.8}$$

$$P_j(s) = \{\theta_j(s), B_j(s), C_j(s), D_j(s), E_j(s)\} \tag{4.9}$$

$$\bar{P}_i = \frac{1}{\Delta s} \int_{s_{i-1}}^{s_i} \bar{P}(s) ds \tag{4.10}$$

4.3 Long-term MPC

The long-term MPC has two primary functions: it prevents the velocity from going below the minimum allowed by speeding up before a section where it will slow down. At the same time, it tries to follow the reference velocity.
The long-term MPC is defined through the following equations. The cost function is defined in equation (4.11), the constraints are shown in equations (4.12), (4.13) and the state function is shown in equation (4.14) where F_{xj} comes from the Pacejka magic formula seen in equation (3.7)

$$J = \int (v_i - v_{\text{ref}})^\top Q (v_i - v_{\text{ref}}) dt \tag{4.11}$$

Subject to:

$$v_{\min} \leq v \leq v_{\max} \tag{4.12}$$

$$\kappa_{\min} \leq \kappa \leq \kappa_{\max} \tag{4.13}$$

$$\dot{v} = \frac{dv}{dt} = \frac{F_{x1}X_1 + F_{x2} + F_{x3} - mg\sin(\theta) - F_{\text{Drag}} - F_{\text{RR}}}{m} \tag{4.14}$$

The control action is κ which is then used as a reference for the wheel slip controller. $\kappa_{\max}$ and $\kappa_{\min}$ are the slip limits, where there is maximum traction and the slip where there is maximum negative traction.
F_z is defined as

$$F_z = m_{\text{axle}} g \cos(\theta)$$

where m_{axle} and g are assumed to be constant. This means that the change in normal force while accelerating or decelerating is assumed to be 0. F_d and F_r are assumed to be constant in order to simplify the problem.

To make the road parameters known at each point in the prediction horizon, the MPC is made to be based on distance instead of time. To switch the MPC from the time domain to the distance domain, the dynamics and cost function are divided by the velocity. This can be shown by (4.15).

$$\frac{dv}{dt} = \frac{dv}{dt}\frac{ds}{ds} = \frac{dv}{ds}\frac{ds}{dt} = \frac{dv}{ds}v \tag{4.15}$$

This gives the following cost function in equation (4.16) and state function from equation (4.17).

$$J = \int \frac{(v_i - v_{\text{ref}})Q(v_i - v_{\text{ref}})}{v_i} ds \tag{4.16}$$

$$\dot{v} = \frac{dv}{ds} = \frac{F_{x1}X + F_{x2} + F_{x3} - mg\sin(\theta) - F_{\text{Drag}} - F_{\text{RR}}}{mv} \tag{4.17}$$

The division by v that happens when changing from time domain to distance domain makes the system more nonlinear compared to a time based model. To remove this nonlinearity, the MPC is switched to modeling the change in kinetic energy using equation (4.18) instead of the change in velocity.

$$E = \frac{1}{2}mv^2 \tag{4.18}$$

$$\frac{dE}{ds} = \frac{dE}{dv}\frac{dv}{ds} = mv\frac{dv}{ds} \tag{4.19}$$

Using (4.17) in (4.19) gives the continuous time cost function (4.20) and the state function (4.21)

$$J = \int (E - E_{\text{ref}})^\top Q(E - E_{\text{ref}}) \tag{4.20}$$

$$\dot{E} = F_{x1}X_1 + F_{x2} + F_{x3} - mg\sin(\theta_i) - F_{\text{Drag}} - F_{\text{RR}} \tag{4.21}$$

To make the values smaller, which helps the solver, the kinetic energy is normalized by the maximum kinetic energy (4.22)

$$\tilde{E} = \frac{E}{E_{\max}} \tag{4.22}$$

After normalization and discretizing the system, the final discrete time MPC formulation is found. The cost function is defined through equation (4.23) where the cost is weighted by the length of the distance step Δs_i. The constraints are (4.24), (4.25) and the state equation is (4.26).

$$J = \sum_{i=0}^{N} (\tilde{E}_i - \tilde{E}_{\text{ref}})^\top Q (\tilde{E}_i - \tilde{E}_{\text{ref}}) \Delta s_i \tag{4.23}$$

Subject to:

$$\tilde{E}_{\min} \leq \tilde{E}_i \leq \tilde{E}_{\max} \tag{4.24}$$

$$\kappa_{\min} \leq \kappa_i \leq \kappa_{\max} \tag{4.25}$$

$$\tilde{E}_{i+1} = \tilde{E}_i + (F_{x1}^{(i)} X_1 + F_{x2}^{(i)} + F_{x3}^{(i)} - mg\sin(\theta_i) - F_d - F_r) \frac{1}{E_{\max}} \Delta s \tag{4.26}$$

One problem with using distance steps is that the distance traveled between control updates is not known. To prevent this from causing any issues, the step distance should be longer than the distance traveled between two updates. This is because a shorter distance step than the distance traveled would cause the MPC to try to use changes in control action that it does not have access to. One way to make the length as accurate as possible is to have a varying step length for the first distance steps in the horizon. This step length is defined in equation (4.27)

$$\Delta s_1 = 1.5 v T_s \tag{4.27}$$

The 1.5 assumes that the distance covered during this time sample will not be more than 50% longer than what would be covered assuming constant speed. When setting up the MPC, the distance step cannot be smaller than the distance traveled in each MPC update.

4.3.1 MPC Solver Configuration

Although `acados` can achieve solve times faster than CasADi+IPOPT for small to medium scale MPC problems, its workflow involves additional steps such as code generation, C-compilation, and template management. Since this thesis does not target real-time or embedded deployment, `CasADi` was chosen because it offers a more streamlined development process. CasADi allows for interactive model definition and easy solver switching. Its shared symbolic layer with `acados` also facilitates future migration to embedded platforms with minimal changes. Therefore, CasADi was chosen for its simplicity, flexibility, and low setup effort.

For the NLP solver, **IPOPT** was chosen due to its robustness and wide applicability. It reliably handles arbitrary nonlinear programs without relying on specific problem structures. While **FATROP** exploits the sequential structure of trajectory optimization problems like MPC and path planning to scale efficiently with increasing problem size, this relatively small-scale problem did not benefit significantly from FATROP's structural advantages in terms of runtime.

Lastly, a **multiple shooting** method was used over single shooting. This was done because multiple shooting led to faster solve times and more stable computation compared to single shooting. Another added benefit was that of full-trajectory

warm-starting, which reduced computation time and improved computational stability.

4.3.2 Including difflock in the MPC

One way to include difflock control in the long-term MPC would be to use a mixed-integer solver, but that has a high computational load. For comparison, using a mixed integer solver (*BONMIN*) on the same system but adding the differential lock as a controllable state yields the following results. With 200 horizon the IPOPT takes 40ms to find the optimal solution. With 40 horizon bonmin takes 197 seconds to find a solution. In mixed integer programming, the solver uses a search tree to find the optimal solution. This means that in the worst-case scenario, the computation time will be multiplied by the number of branches and most likely scale exponentially. For the problem with the differential lock, it scales 2^N, but if it is simplified to only be allowed to be turned on or off once, then it will scale by N which will lead to long computation times for problems with long horizons.

To combat this problem of the differential lock being an integer value, it can instead be represented as a continuous variable between 0 and 1. This can be done through the function (4.28).

$$\mathbf{X}(\sigma_x) = \frac{1}{2}\Big[\tanh\Big(\alpha\,(\sigma_x - c)\Big) + 1\Big] \;+\; 0.05\,(\sigma_x - c). \tag{4.28}$$

When the variable σ_x exceeds or falls below a certain threshold c, the function maps to a value close to zero or one, effectively indicating whether the differential lock should be engaged. To discourage unnecessary usage, the cost associated with activating the differential lock is kept high, prompting the MPC to activate it only when essential. However, this setup can lead to situations where the differential lock signal remains in an intermediate state, leading to a degradation in its accuracy. Additionally, the previous approach lacked a mechanism to prevent frequent switching of the differential lock in regions with high uncertainty, which often resulted in chattering behavior in the control output. To address this limitation, it was replaced by a dedicated controller focused solely on differential lock purposes.

4.4 Wheel Speed Controller

Two different wheel speed controllers were proposed to control the wheel speed. Both were able to control it, and the performance differences are further discussed under results. The wheel dynamics that the controller should control are the ones shown in equation (3.6)
To linearize the system, a feedforward part was added, which gives the control law in equation (4.29)

$$\tau = u + F_{x1} r X_1 + F_{x2} r + F_{x3} r \tag{4.29}$$

Which gives the linearized system (4.30).

$$\frac{d\omega}{dt} = \frac{u}{I_1 X_1 + I_2 + I_3} \tag{4.30}$$

where u is the output from the wheel speed controller. In the first version of the controller, F_{xj} was calculated based on the current slip and road parameters. This improved the performance of the controller, but the estimates of the road surface and the slip will always have some inaccuracies.

To improve response and reduce the effect of inaccuracies, the feedforward part was changed to use the reference slip instead of the actual slip to calculate F_{xj} used in equation (4.29). For the feedforward part, F_{xj} is calculated using equation (4.31). This causes the system to immediately get the torque needed to reach the operating point, and then the wheel speed controller tries to keep it there.

$$F_{xj} = F_{zj} D sin[C \arctan\{B\kappa_{\text{ref}} - E(B\kappa_{\text{ref}} - \arctan(B\kappa_{\text{ref}}))\}] \tag{4.31}$$

Based on the reference slip and the current velocity, a reference angular velocity ω_{ref} is calculated for the wheel speed controller to follow according to equation (4.32)

$$\omega_{\text{ref}} = \frac{v}{r(1 - \kappa_{\text{ref}})} \tag{4.32}$$

4.4.1 PI controller

For a control law to combine with the feedforward part, a PI controller was used. The PI controller has the control law from equation (4.33) that works based on the difference between the angular velocity and the reference angular velocity.

$$u = -K_p(\omega - \omega_{\text{ref}}) - K_i \int (\omega - \omega_{\text{ref}}) \tag{4.33}$$

4.4.2 Unconstrained MPC

Unconstrained MPC for the wheel speed controller is used to leverage the predictions from the long-term MPC. The optimal control action for linear unconstrained MPC can be calculated through a batch approach [14]. This gives a low enough computation time to make it reasonable for the fast acting wheel speed controller. The predicted velocities are used to show the wheel speed controller how the reference will change over the prediction horizon. To transform how the velocity changes over distance to changes over time, constant acceleration a over the distance step is assumed for the long-term MPC. Since the sampling time is fast compared to the distance step length, it can be assumed that the entire prediction horizon will be within one distance step. This will hold true as long as the velocity is not close to 0 ($v < 0.5m/s$)

$$a = \frac{v_{i+1}^2 - v_i^2}{2\Delta s_i} \tag{4.34}$$

To get the velocity at each time step in the unconstrained MPC, the velocity comes from (4.35) from which the reference angular velocity is calculated according to (4.32)

$$v_i = v_0 + aT_s \cdot i \tag{4.35}$$

The linearized system (4.30) is used to create a discrete time state space model witch is (4.36)

$$A = 1, \quad B = \frac{1}{I_1 X_1 + I_2 + I_3} T_s \tag{4.36}$$

The cost function used is found in equation (4.37) using the weight matrices found in equation (4.38). R is structured from two parts, D which is a penalty added to change in control input. It is multiplied with its transpose to have the cost relating to the square of the change in control input. α is a scaling factor to scale the cost of change in control input.

$$J = \sum_{i=1}^{N} (\omega_i - \omega_{\text{ref}}) Q (\omega_i - \omega_{\text{ref}}) + u_i R u_i \tag{4.37}$$

$$Q = I, \;\; R = \alpha D^\top D, \;\; D = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ -1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & -1 & 1 \end{bmatrix} \tag{4.38}$$

The control input for the unconstrained control problem is found by equation (4.39)

$$\boldsymbol{u}^*(k) = -(\Gamma^\top \bar{Q} \Gamma + \bar{R})^{-1} \Gamma^\top \bar{Q} \Omega x(k) \tag{4.39}$$

Ω and Γ can be found through equation (4.40)

$$\Omega = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \;\; \Gamma = \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix} \tag{4.40}$$

$$x(k) = \omega - \omega_{\text{ref}}(k) \tag{4.41}$$

5

Results

5.1 Comparison of Wheel Speed Controllers

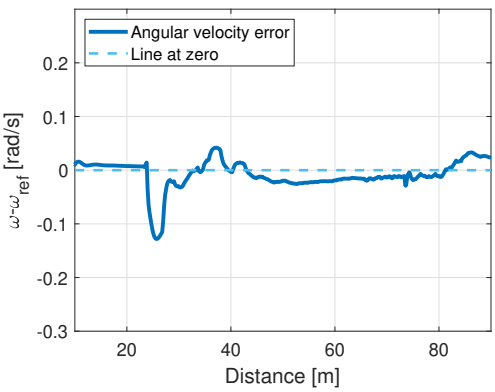
The wheel speed controllers are compared to each other through two different plots. In this section, Wheel Speed Controller 1 (WSC1) refers to the controller using the control law from Equation (4.33) and WSC2 refers to the controller using the control law from Equation (4.39). First, there is a plot on how the error between the reference angular velocity and the measured angular velocity changes over time. This plot shows how the controller behaves when there is an error. The other plot is how the torque changes over the simulation. This shows if the controller gives too fast changes in torque, which might not be realistic in a real vehicle.

Controller name	**RMSE (comp. time)**	**Max (error)**
WSC1 with reference slip feedforward	0.0280	0.1282
WSC2 with reference slip feedforward	0.0649	0.1998s
WSC1 with linearizing feedforward	0.0876	0.2403
WSC2 with linearizing feedforward	0.1028	0.2246
WSC1	0.1288	0.2749
WSC2	0.1354	0.2762
PI with reference slip feedforward 20ms	0.0498	0.1921

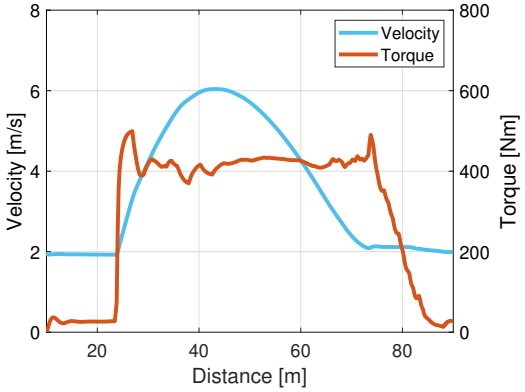
Table 5.1: Comparison of tracking performance of different wheel speed controllers

5.1.1 Reference slip based Feedforward Wheel Speed Controller comparison

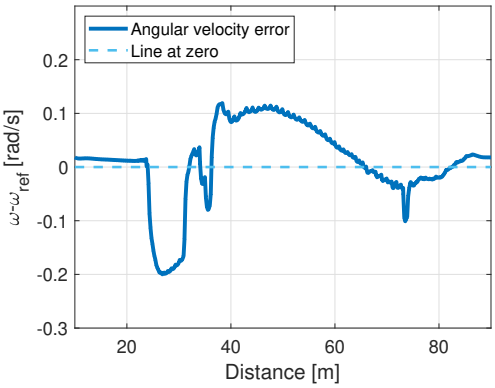
In Figure 5.1 WSC1 is compared to WSC2 where both of them also have a feedforward part based on the reference given by the long-term MPC. Figure 5.1a and Figure 5.1c show that both controllers follow the reference well, but WSC1 has a smaller error than WSC2. Looking at the torque in Figure 5.1b and Figure 5.1d shows that it is similar between them except for small oscillations for WSC2.



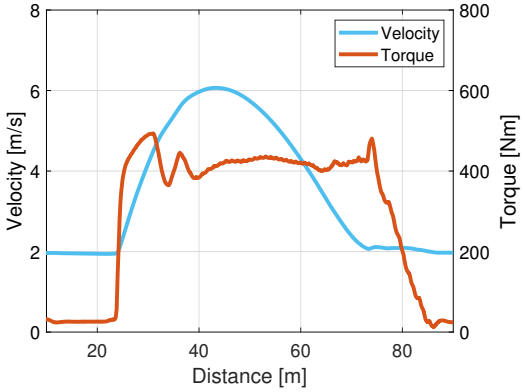
(a) WSC1: Error between actual and reference angular velocity



(b) WSC1: Torque and velocity



(c) WSC2: Error between actual and reference angular velocity



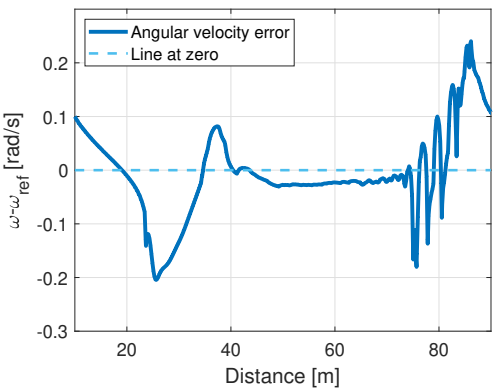
(d) WSC2: Torque and velocity

Figure 5.1: Wheel speed controller behavior under varying incline (peak 20%), $\mu = 0.2$, and reference velocity of 2 m/s. Top: WSC1 controller with reference feedforward gain. Bottom: WSC2 with reference feedforward gain.

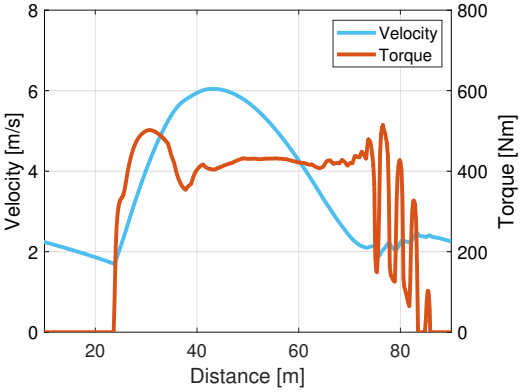
5.1.2 Linearizing Feedforward Wheel Speed Controller comparison

For the case in Figure 5.2 the feedforward part is based on the measured slip. This part has a problem in that there is always a mismatch between the tire model and the real tire dynamics. This makes it so that for some scenarios it might be difficult for the controller to keep the slip constant as the feedforward part might work against the controller. In Figure 5.2a and Figure 5.2c some oscillatory behavior can be observed at the end of the simulation. This is most likely caused by a mismatch between the real tire dynamics and the tire model used in the linearizing feedforward calculation. Even when disregarding the oscillatory behavior, it performs worse than the feedforward controller that is based on the reference slip. Looking at the torque in Figure 5.2b and Figure 5.2d it can be seen that the torque is similar to the case in Figure 5.1 except for the oscillatory part. This oscillatory part could be removed

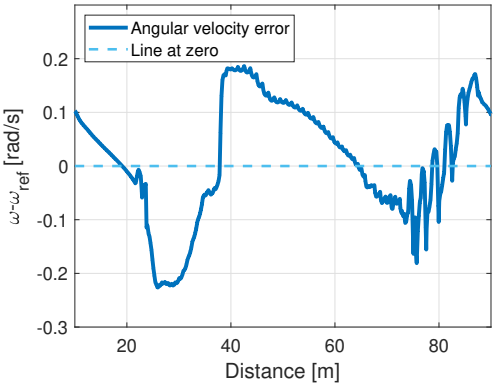
by scaling down the linearizing feedforward part, but that would get a behavior that is not as fast responding and more similar to the one without linearizing feedback.



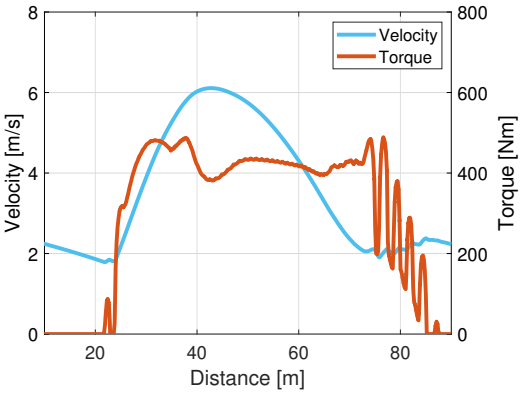
(a) WSC1: Error between actual and reference angular velocity



(b) WSC1: Torque and velocity



(c) WSC2: Error between actual and reference angular velocity

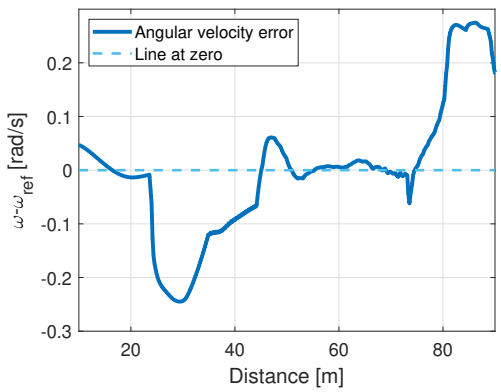


(d) WSC2: Torque and velocity

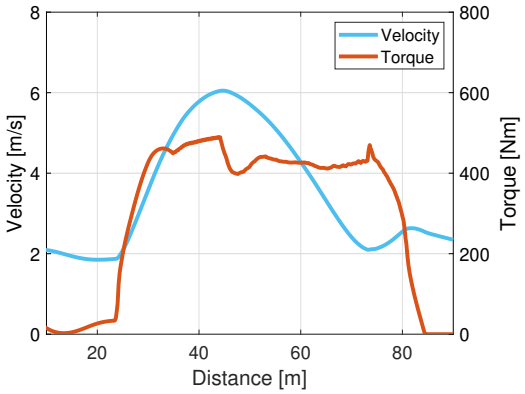
Figure 5.2: Wheel speed controller behavior under varying incline (peak 20%), $\mu = 0.2$, and reference velocity of 2 m/s. Top: WSC1 with linearizing feedforward gain. Bottom: WSC2 with linearizing feedforward gain.

5.1.3 Wheel Speed Control without Feedforward

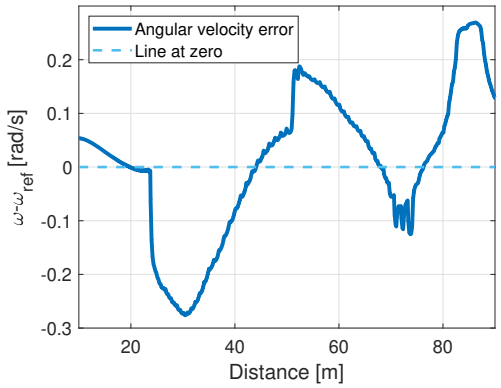
Figure 5.3 shows the simulation for scenarios without any feedforward part on the wheel speed controller. Here it can be seen that both WSC1 and WSC2 behave almost identically. They both have a larger maximum error and a larger average error, which gives a slower response compared to having a feedforward part.



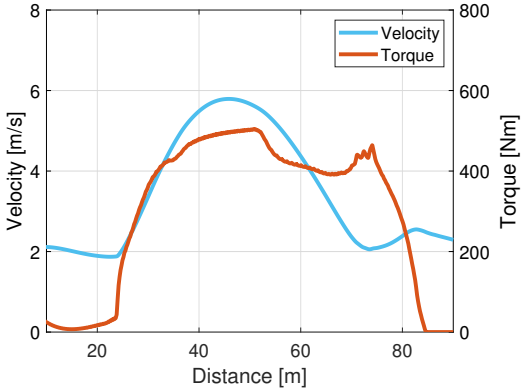
(a) WSC1: Error between actual and reference angular velocity



(b) WSC1: Torque and velocity



(c) WSC2: Error between actual and reference angular velocity

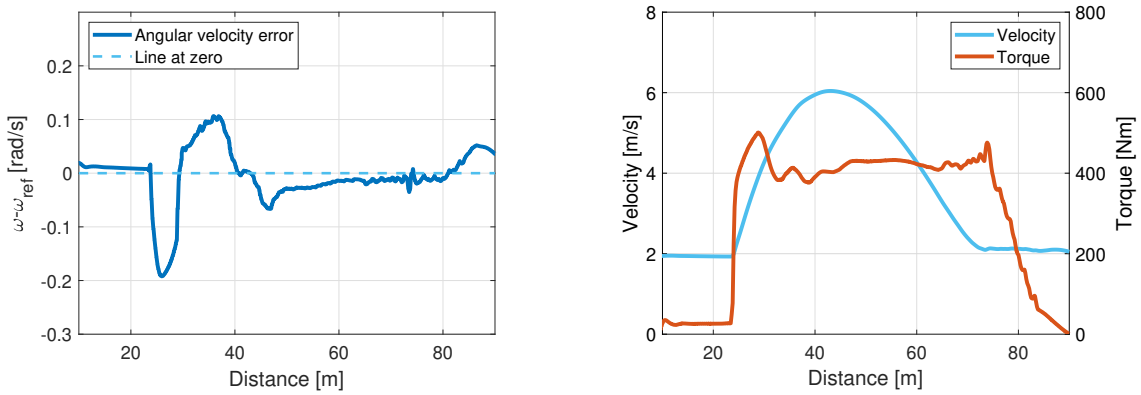


(d) WSC2: Torque and velocity

Figure 5.3: Wheel speed controller behavior under varying incline (peak 20%), $\mu = 0.2$, and reference velocity of 2 m/s. Top: WSC1. Bottom: WSC2.

5.1.4 Effect of Sampling time

The wheel speed controller was tested at three different sampling times: 2ms, 20ms and 200 ms. The tests were made using WSC1 with a feedforward gain based on reference slip as seen in Figure 5.1a. Comparing Figure 5.1a, Figure 5.4 shows that there is not that big of a difference in performance between $2ms$ and $20ms$ sampling time, but for the case with $200ms$ the simulation did not finish as the truck could not climb the hill for this test case. But this is a test case where the road surface is constant. When the road surface is varying, then the lower sampling times have difficulty reacting to changes in road surfaces and the velocity goes to zero.



(a) Error between measured and reference angular velocty

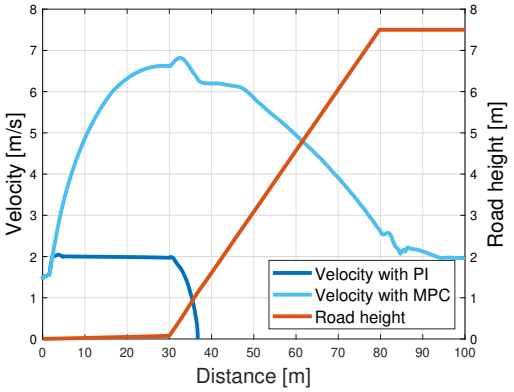
(b) Velocity and torque

Figure 5.4: Tracking performance for wheel speed controller with sampling time of 20ms

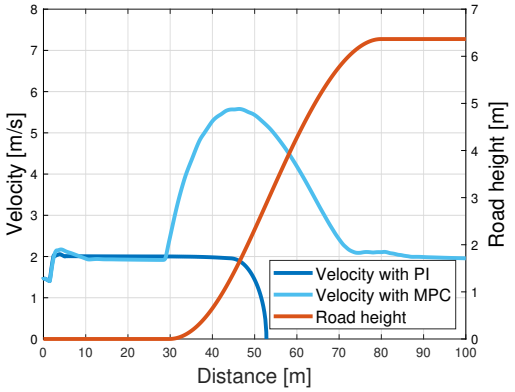
5.2 Comparison of MPC to Reference-Following PI Control

To demonstrate how the long-term MPC enables the vehicle to climb hills that it couldn't without assistance, its performance is compared to that of a PI controller tasked with following the reference velocity. This PI controller works on the difference between the measured velocity and the reference velocity and outputs a reference slip that the lower level controller follows. All the tests in this section are done with both controlled differentials as open. This is to only show the difference between the MPC and the PI reference.

In figure 5.5a it can be seen that without MPC, the vehicle starts slowing down immediately after the incline starts and the velocity goes to zero. For the case using MPC, the vehicle speeds up before the incline, and it has enough kinetic energy to clear the hill. Figure 5.5b shows that for a varying slope, both the PI and MPC slow down at the steepest part. The MPC has predicted the loss of velocity and could therefore compensate for it by speeding up beforehand.



(a) Speed control comparison using PI and MPC on a 15% constant slope, $\mu = 0.2$, reference velocity = 2 m/s.



(b) Speed control comparison using PI and MPC on a variying slope with maximum 20% , $\mu = 0.2$, reference velocity = 2 m/s.

Figure 5.5: Comparison between a PI controller and MPC

Figure 5.6 shows the difference between a controller that follows the reference speed and a controller using MPC. In this test only the two rear axles are driven, and below the figure it can be seen when each axle encounters the slippery patch. Here it can be seen that the MPC speeds up before the slippery patch, so it has enough kinetic energy to clear that part. The PI controller that only follows a reference speed does not speed up and therefore cannot climb the hill.

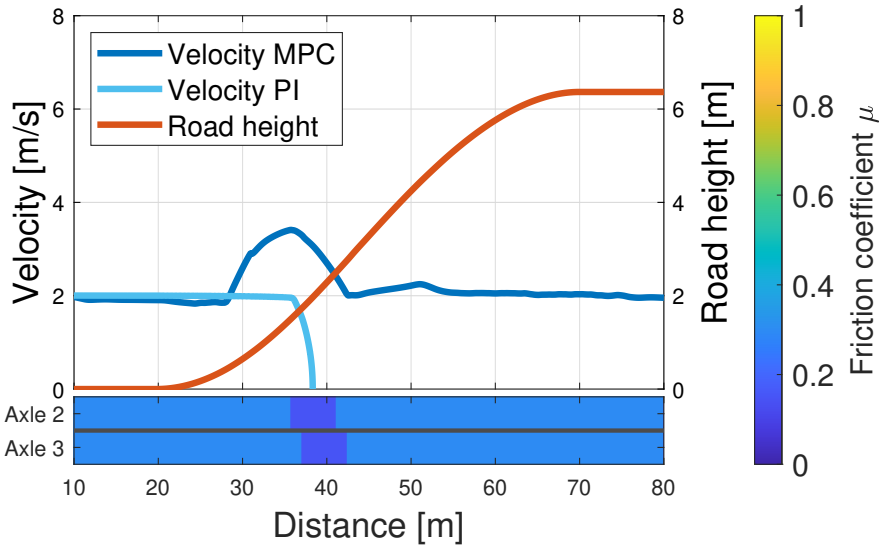


Figure 5.6: Speed control comparison using PI and MPC on a varying slope with maximum 20% .μ is 0.3 except for a 10 meter part of 0.15. reference velocity is 2 m/s.

Figure 5.7 shows how the PI controller compares to the MPC when the road surface is varying a lot. Here it can be seen that the MPC can climb the hill, but the velocity is not as smooth as in the other test cases. The PI controller cannot climb the hill in this scenario.

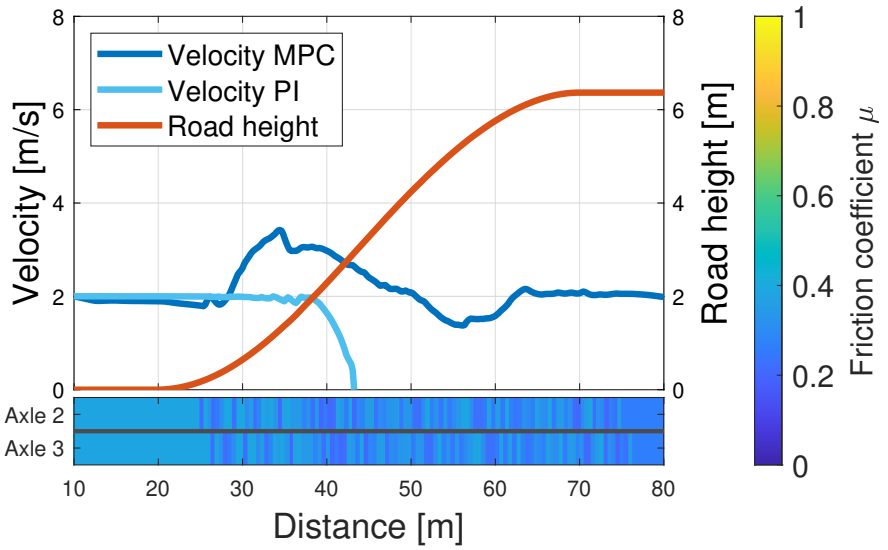


Figure 5.7: Speed control comparison using PI and MPC on a varying slope with maximum 20%. μ is random at each meter in the distance uniformly distributed between 0.2 and 0.4. The reference velocity is 2 m/s.

5.3 Computation Time of MPC

The current hardware configuration comprises a 12th Gen Intel® Core™ i7-12850HX processor operating at 2.10 GHz, coupled with 32.0 GB of RAM. The figure 5.8 presents the minimum, mean, and maximum computation times across various prediction horizons. Notably, the mean computation time exhibits a linear increase with the extension of the prediction horizon. This shows that even longer horizons could be used if there is more compute power available.

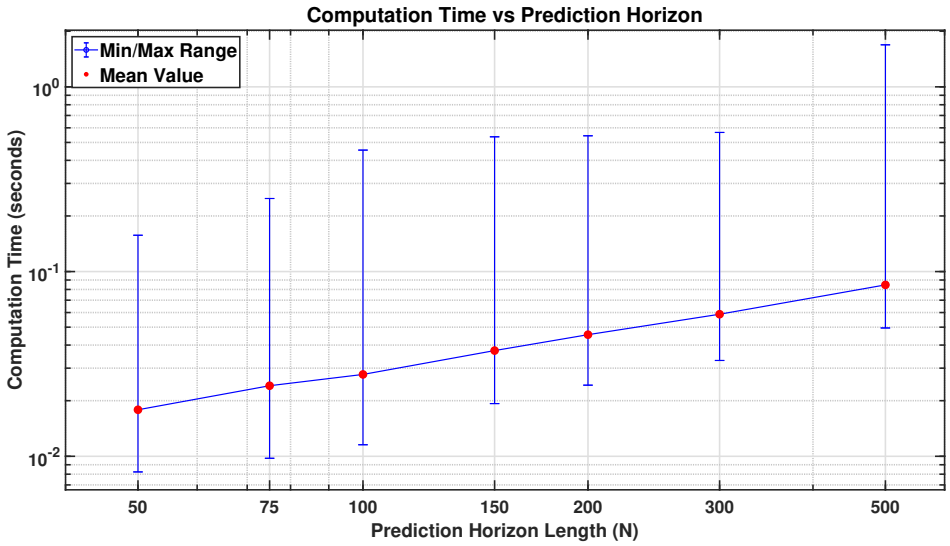


Figure 5.8: Comparison of computation time for long term MPC with different horizon lengths. The simulation was done on a slope of 15% and $\mu = 0.2$

5.4 Robustness

The robustness of the controller is tested by adding noise and delay to the vehicle measurements. By adding noise to the future road information to make it more realistic, and also modeling the response of the engine as a delay or transfer function.

5.4.1 Engine Delay

The transfer function used to model the delay of the engine is a first-order plus dead time model, as seen in equation 5.1.

$$G(s) = \frac{e^{-Ls}}{t_c s + 1} \tag{5.1}$$

According to [19] typically $L = 20ms$ is a standard delay between control input and engine response. This dead time is tested with different time constants t_c to show how the delay affects the performance. In Figure 5.9 it can be seen that both the case with $t_c = 0.1$ and with $t_c = 0.2$ perform similarly to the system without delay. This shows that the controller can handle a delay of $20ms$ and with a time constant of $200ms$ in at least this test case. The system has more overshoot while first converging to the reference when there is a larger time constant. For the case with a time constant of $500ms$ then it cannot climb the hill. It is forced to turn on both inter-axle locks a bit before 40 meters, but it is still not enough to climb the hill with this delay.

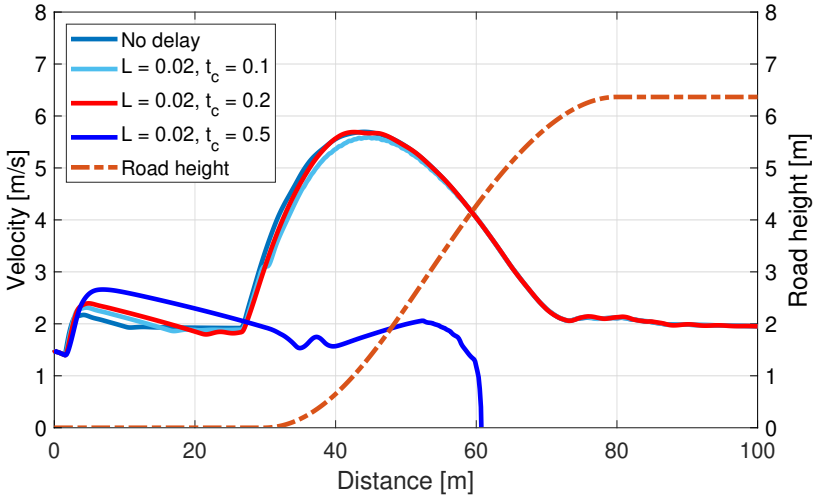


Figure 5.9: Test of the effect on the system response for different delays. Variable incline with maximum 20%. $\mu = 0.2$, $v_{\text{ref}} = 2$

5.4.2 Predicted data

To test the performance with inaccurate road information, the controller was tested using road data that differed from the data used in the simulation. Noise was added to the data used in the controller. The noise is normally distributed with a standard deviation that is a percentage of the real value. In Figure 5.10 it can be seen that even with bad road information it can still climb the hill. It is important to note that the mean is still the same as without noise. The reason for the vehicle having higher velocity when there is more noise is that the rear interaxle difflock is not engaged here. This means that the MPC bases its calculations on the wheel with the least traction. When the input data has higher variance, there's a greater chance that the predicted minimum future traction will be underestimated, leading the controller to apply a more conservative strategy, even if the actual available traction is higher.

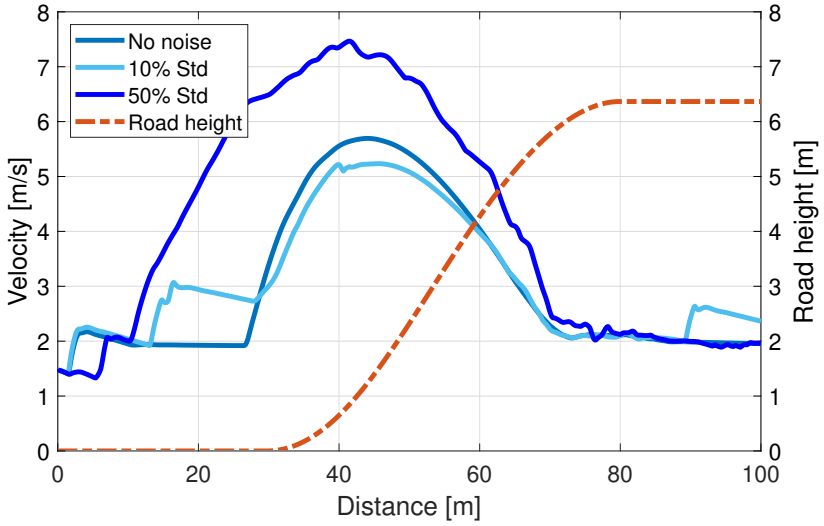


Figure 5.10: Test of the effect on the system response for different noise levels on the predicted tire information. Variable incline with maximum 20%. $\mu = 0.2$, $v_{\text{ref}} = 2$

6
Conclusion

6.1 Research Questions

In this thesis, a traction controller using MPC and road information was developed. It was observed that the computation time of the MPC was too long to handle traction using road information. This was solved by splitting the controller into two parts, where a low frequency NMPC handles the long term predictions and a wheel speed controller handles the wheel spin. This approach showed promising results as both sub-controllers could be tested and changed individually. For the wheel speed controller, an unconstrained MPC with a feedforward part was developed that could control traction. A PI controller with a linearizing part was also developed, and according to testing, both of the controllers were effective. The PI controller still performed a bit better than the unconstrained MPC. The road information was used in the long term NMPC which in testing is shown to be able to climb hills that would not be possible without it.

The system developed combines traction control, velocity planning and differential lock control into a single system. It was done through creating one controller for each problem. Combining all three parts into a single controller could be done, but the computation time would be to high compared too the necessary sampling time. The control system was robust in that it could handle both delays and measurement noise.

6.2 Future work

Future developments of this research could involve incorporating lateral dynamics and integrating the system with other control modules to enable deployment in real vehicles. The long term MPC could have enhanced realism by adding a way to handle road surfaces that are different for the left and right sides of the vehicle. Finding a way to describe rough surfaces would be another added realism for the controller, as now it assumes a smooth surface and roughness would reduce traction. Combining it with controllers for lateral stability and rollover prevention could be another way to further develop this system. This could be done through adding stricter slip constrains for the long term MPC to improve lateral stability, and this slip constrain then comes from a system that controls lateral stability. The same thing could be done for rollover prevention but instead limiting the maximum velocity to prevent rollover.

Bibliography

[1] Rajamani, R. (2012). *Vehicle Dynamics and Control* (2nd ed.). Springer, New York, NY, USA.

[2] Scamarcio, L., So, J., Lenzo, B. (2023). Predictive anti-jerk and traction control for V2X connected electric vehicles with central motor and open differential. *IEEE Transactions on Vehicular Technology*, 72(6), 7365–7377. https://doi.org/10.1109/TVT.2023.3265903

[3] Tavolo, G., So, K. M., Tavernini, D., Perlo, P., Sorniotti, A. (2024). Nonlinear model predictive control for preview-based traction control. *arXiv preprint arXiv:2406.02206*. Retrieved from `https://arxiv.org/abs/2406.02206`

[4] Li, H.-Z., Li, L., He, L., Kang, M.-X., Song, J., Yu, L.-Y., & Wu, C. (2012). PID plus fuzzy logic method for torque control in traction control system. *International Journal of Automotive Technology*, 13(3), 441–450. https://doi.org/10.1007/s12239-012-0041-4

[5] Kuntanapreeda, S. (2014). Traction control of electric vehicles using sliding-mode controller with tractive force observer. *International Journal of Vehicular Technology*, 2014, Article ID 829097, 10 pages. https://doi.org/10.1155/2014/829097

[6] Palma, Á., Reyes, A., Rohten, J., Risso, N., Quezada, D., & Esparza, V. (2021). MPC-based traction control for electric vehicles. In *Proceedings of the 2021 IEEE International Conference on Automation/XXIV Congress of the Chilean Association of Automatic Control (ICA-ACCA)* (pp. 1–6). IEEE. https://doi.org/10.1109/ICA-ACCA54658.2021.9465944

[7] Jalali, M., Khajepour, A., Chen, S.-K., & Litkouhi, B. (2016). Integrated stability and traction control for electric vehicles using model predictive control. *Control Engineering Practice*, 54, 256–266. https://doi.org/10.1016/j.conengprac.2016.06.005

[8] Tavernini, D., Metzler, M., Gruber, P., Sorniotti, A. (2019). Explicit Nonlinear Model Predictive Control for Electric Vehicle Traction Control. *IEEE Transactions on Control Systems Technology*, 27(4), 1438-1451. https://ieeexplore.ieee.org/document/8390933

[9] Joel A. E. Andersson, Joris Gillis, Greg Horn, James B. Rawlings, and Moritz Diehl. CasADi—A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019. 10.1007/s12532-018-0139-4.

[10] United Nations. (2015). *Goal 3: Ensure healthy lives and promote well-being for all at all ages*. Retrieved from `https://sdgs.un.org/goals/goal3`

[11] United Nations. (2015). *Goal 9: Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation.* Retrieved from `https://sdgs.un.org/goals/goal9`

[12] United Nations. (2015). *Goal 11: Make cities and human settlements inclusive, safe, resilient and sustainable.* Retrieved from `https://sdgs.un.org/goals/goal11`

[13] Fröjd, N. (2021). *Handling Analysis and Control Development of Commercial Trucks with Volvo Transport Models*, *MathWorks*. Retrieved from mathworks website

[14] Murgovski, N. (2024). Model Predictive Control Lecture Notes. *Chalmers University of Technology*

[15] Verschueren, R., Frison, G., Kouzoupis, D., Frey, J., van Duijkeren, N., Zanelli, A., Novoselnik, B., Albin, T., Quirynen, R., & Diehl, M. (2022). acados—a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, 14(1), 147–183. `https://doi.org/10.1007/s12532-021-00208-8`

[16] Wächter, A., & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1), 25–57.

[17] Vanroye, L., Sathya, A., De Schutter, J., & Decré, W. (2023). FATROP: A fast constrained optimal control problem solver for robot trajectory optimization and control. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 10036–10043. IEEE

[18] Pacejka, H. B. (1986). The magic formula tire model. *Vehicle System Dynamics*, 15(S1), 1–18.

[19] Grünbacher, E., del Re, L., Kokal, H., Schmidt, M., Paulweber, M. (2008). Adaptive control of engine torque with input delays. In *Proceedings of the 17th World Congress of the International Federation of Automatic Control (IFAC)*, Seoul, Korea, July 6–11, 2008 (Vol. 41, No. 2, pp. 9479–9484). Elsevier. https://doi.org/10.3182/20080706-5-KR-1001.01602

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY