CHALMERS
UNIVERSITY OF TECHNOLOGY

Implementation of Optimal Energy Management of High Capacity Vehicles with Electrically Propelled Dolly Under Lateral Constraints using CasADi

Master's thesis in Systems, control and Mechatronics

Wilhelm Johannesson
Ying Li

Department of Mechanics and Maritime Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Master's thesis 2022

Implementation of Optimal Energy Management of High Capacity Vehicles with Electrically Propelled Dolly Under Lateral Constraints using CasADi

Wilhelm Johannesson
Ying Li

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
Gothenburg, Sweden 2022

Implementation of Optimal Energy Management of High Capacity Vehicles with Electrically Propelled Dolly Under Lateral Constraint in CasADi
WILHELM JOHANNESSON
YING LI

© WILHELM JOHANNESSON, YING LI, 2022.

Supervisor: Toheed Ghandriz, Volvo Group Truck Technology and Chalmers
Examiner: Bengt Jacobson, Mechanics and Maritime sciences

Master's Thesis 2022:63
Department of Mechanics and Maritime sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: An A-double combination vehicle with a diesel truck and a trailer dolly. The picture is taken from Transportation Mission-Based Optimization of Heavy Combination Road Vehicles and Distributed Propulsion, with the authors approval.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2022

Implementation of Optimal Energy Management of High Capacity Vehicles with Electrically Propelled Dolly Under Lateral Constraint in CasADi
WILHELM JOHANNESSON
YING LI
Department of Mechanics and Maritime sciences
Chalmers University of Technology

Abstract

To lower the environmental impact of road freight transport, high-capacity transport vehicles with electrically propelled dolly trailers (e-dollys) are proposed as a more efficient way of transportation. This distributed propulsion hybrid vehicle introduces challenges regarding lateral stability and new possibilities for energy management. In this thesis, these properties are addressed by a model predictive control (MPC) strategy. The thesis includes design of an optimal energy management MPC for a A-double truck with an e-dolly, using the optimization toolbox CasADi. The MPC controller finds the optimal speed, soc, internal combustion engine (ICE) gears, ICE propulsion, electric motor (EM) propulsion and retardation and braking.

The MPC is designed using CasADis symbolic framework and sequential linear and quadratic programs (SLP) and (SQP), and IPOPT as optimization algorithms. The two dynamic models are: an explicit ordinary differential equation (ODE) describing the longitudinal dynamics and an implicit differential algebraic equation (DAE) describing both lateral and longitudinal dynamics. Discretization of explicit and implicit model dynamics is done with Runge-Kutta and IDAS numerical integration methods. A method for decoupling ICE gears from the mixed integer nonlinear program (MINLP) is designed and gear optimization is done with a one-step algorithm. A method for implementing online shrinking horizon as the truck is getting close to its destination is developed for the sequential programs. Two types of lateral stability constraints are tested, one rule based calculated speed limit, and one directly constraining states in the lateral dynamics model.

The MPCs are evaluated with simulations and the fuel consumption is compared to a diesel truck. The MPC computation time is evaluated in respect to real time implementation. The two lateral stability constraints are evaluated. The MPC using SLP, and the rule based lateral stability constraint is determined to be fast enough to be implemented in real time but, the rule-based stability constraint is lacking and cannot guarantee lateral stability unless very conservative constraints are considered. The MPC including the nonlinear lateral dynamics is too computationally expensive to implement in real time for long horizons, but its optimal trajectories can serve as a benchmark for further simplifications of the lateral dynamics.

keywords: Model predictive control, optimal energy management, motion control, optimization, lateral stability, electrified propulsion, high capacity transport vehicles, distributed propulsion and simulations.

Acknowledgements

Firstly, we would like to express our sincere thanks to our supervisor Toheed Ghandriz and our examiner Professor Bengt Jacobson. They have always provided us with tremendous help using their professional expertise and enthusiasm, which are present throughout the entirety of our project. Also, we appreciate the Volvo Group's assistance in providing us with the opportunity and challenge to complete the project, especially for the comfortable environment and facilities they offered for us.

We would also like to express the special thanks to our beloved friends and parents for their support and companionship. They give us the confidence to confront the unknown and difficulties, even in trying circumstances.

Wilhelm Johannesson and Ying Li, Gothenburg, September 2022

List of Acronyms

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

COG	Center of Gravity
DAE	Differential Algebraic Equations
EM	Electric Motor
HCT	High Capacity Transport
ICE	Internal Combustion Engine
LCV	Long Combination Vehicle
LP	Linear Program
MILP	Mixed Integer Linear Program
MINLP	Mixed Integer Nonlinear Program
MPC	Model Predictive Control
NLP	Nonlinear Programs
NMPC	Nonlinear Model Predictive Control
NOCP	Nonlinear Optimal Control Problem
OCP	Optimal Control Problem
ODE	Ordinary Differential Equations
QP	Quadratic Program
RHC	Receding Horizon Control
RMS	Root Mean Square
RTI	Real-Time Iteration
SLP	Sequential Linear Program
SQP	Sequential Quadratic Program
SOC	State Of Charge

Contents

List of Figures

List of Tables

1
Introduction

1.1 Background

Carbon dioxide emission and energy consumption are estimated to rise with the increased demand for road freight transportation [1]. In 2019, road freight transit accounted for 76.3% of total inland tonne-kilometers of freight transport in Europe [2]. The transportation industry is facing significant challenges in terms of lowering energy consumption and limiting environmental effects. Consequently, there is considerable interest in enhancing the efficiency of transportation systems. High Capacity Transport (HCT) vehicles, i.e., heavier and longer vehicle combinations, are proposed as one of the solutions to this problem. It is established that HCT road vehicles can be up to 13% more energy efficient [3] [4], allowing them to be utilized in an increasing number of nations. Towing units intended for HCTs, on the other hand, will be overpowered if they are utilized in non-HCTs or lightly loaded HCTs. One of the proposed solutions to this can be described as scaling the propulsion for HCTs by adding electrically propelled dollies. A traditional dolly is a non-powered vehicle trailer that is attached to a truck or tractor and can be added or removed depending on the quantity of cargo. In this thesis, the electricly propelled dolly is referred to as an e-dolly. Including a e-dolly in a HCT lowers the power demand of the towing unit and makes the combination vehicle modular. The e-dolly also makes the HCT into a distributed propulsion hybrid vehicle. The distributed propulsion introduces new challenges regarding lateral stability, and the hybrid powertrain introduces new challenges for energy management and possibilities for further improvements in energy efficiency. Both of these challenges can be addressed with a new vehicle control strategy.

Previous research [5] has established that adding an electrically propelled dolly to an HCT vehicle with a diesel-powered towing unit can save 10 to 20 percent energy. However, there are various aspects to consider when using electrically propelled dollies in HCTs. A predictive energy management controller that coordinates the activation of the dolly and towing unit needs to be incorporated in order to achieve enhanced performance with regards to energy consumption. In [6], several model predictive controllers are derived for optimal energy management, and their controllers are fast enough to be implemented in real-time in an HCT vehicle. However, in [7], it is shown that the lateral stability will be reduced while applying these controllers on long combination vehicles.

The worst case scenario related to lateral instability occurs when the energy management controller predicts that it is more energy efficient to only uses the electric drive axle for braking and propulsion. By only using the electric drive axle for braking, the combination vehicle is at risk of jack-knifing around the e-dollys articulation points, as is shown in [7], this scenario discussed in detail in 3.2.4. Only using the electric axle for propulsion can similarly cause the e-dolly to push the front unit sideways. Theses kind of situations can arise on curvy roadways with elevation changes and poor road grip and uneven distribution of load between axles. Because of these risks it is essential for the controller to evaluate the trade-off between safety and optimal energy management.

The strategies for handling lateral stability stated in [7] are implemented, the strategies are based on either including a rule based speed constraint or including the vehicle lateral dynamics and tire model as constraints in the model predictive controller. The vehicle lateral dynamics model are developed based on the methods described in [8]. The lateral dynamics are complex and highly nonlinear, resulting in a much computationally expensive model predictive controller. Meanwhile, several strategies in [7] for simplifying the constraints are presented, with the goal of improving the computational efficiency while still accounting for lateral stability.

1.2 Purpose

The purpose of this thesis is to design a energy optimal model predictive controller using CasADi that accounts for lateral stability, and can be implemented in real time. The controller will be evaluated in simulations and the resulting fuel consumption is compared to a simulated conventional diesel vehicle. The driving scenario is a hilly and curvy country road with no interruptions such as traffic signals or traffic jams. The model parameters are based on an A-double diesel heavy combination vehicle with an electric dolly.

1.2.1 Objectives

The purpose of this thesis is gathered into 3 specific objectives that are described as follows:

- Design a model predictive controller that finds the optimal propulsion distribution, speed, battery state of charge, and gear trajectories for the hybrid combination vehicle, using CasAdi.
- Implement lateral stability constraints.
- Evaluate controller performance in regards to fuel consumption, real time implementation and lateral stability.

1.3 Organization

The first chapter of the thesis contains the introduction, including the background, purpose and objectives. The rest of this thesis is organized as follows: Chapter 2

explains the main theories applied in the thesis; Chapter 3 presents the methods which are implemented in solving the motion control and energy management problem; Chapter 4 shows the results using different types off MPCs and two different constraints to ensure lateral stability; Chapter 5 summarizes the conclusion drawn from the results and suggestions on futher work on the this topic. Finally, a list of references are attached in the end.

2

Theory

2.1 Model predictive control (MPC)

Model predictive control (MPC) is an advanced model-based control method that finds the optimal control signals for a limited future horizon. MPC utilizes a dynamic model to predict the behavior of a closed loop system and has been widely adopted for vehicle-related control problems.

In an MPC controller, the control signal is generated by solving an optimization problem with a cost function that represents the desired behavior of the controlled system over a finite horizon, N. The optimization problem can be extended with constraints that represent physical limitations on control signals and states, as explained in [26]. These optimization problems can also be refereed to as optimal control problems (OCP) and if the optimization problem is non linear, NOCP. MPC is a discrete control method and each step in the discrete control horizon is refereed to as a stage, k to $k + N$ in figure 2.1.

In a closed-loop system with a MPC controller the control signals and the states in the horizon are recalculated as the system increments forward and only the current inputs are fed to the plant , this is refereed to as moving horizon. Recalculating the prediction horizon at each step makes the controller more robust with respect to model errors and disturbances. A schematic figure of the states and input trajectory of a MPC with moving horizon is shown in figure 2.1. For small control problems the calculated horizon can be discarded after the inputs have been past to the plant, this is refereed to as receding horizon (RHC). For large or complicated optimal control problems that are computationally expensive the solution of the current prediction horizon can be reused as an initial guess for the optimal control problem in the next stage, this is refereed to as warm starting the optimization algorithm or solver, this speeds up the computation time at the price of passing around and saving data.

For MPCs using sequential programs, described in 2.2.1 there is an method to speed up the computation times, at the price of a less optimal solution. This method relies on warm starting the algorithm with its previous solution and then forcing it to stop after a fixed number of iterations. This method can be referred to as a simplified version of the, real time iterations or (RTI) algorithm, further described in [7] and [28].

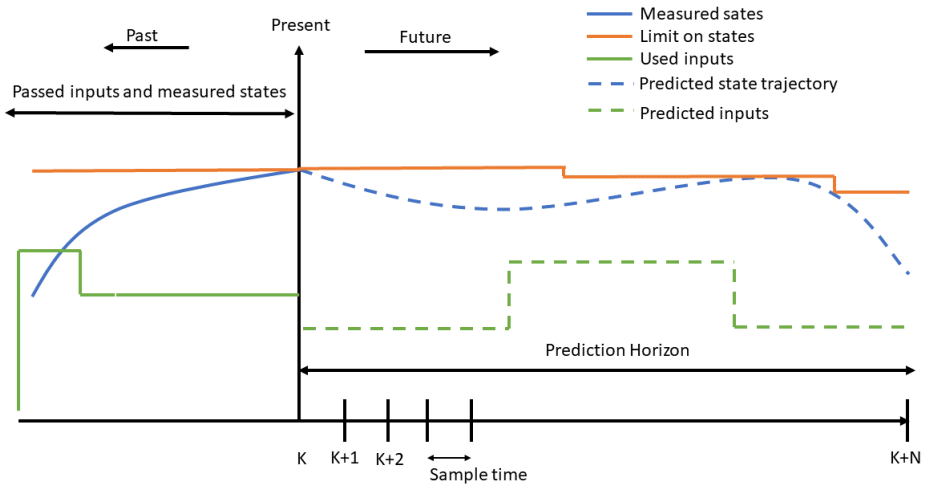

Figure 2.1: Schematic figure of the involved parts of a MPC controller. As the controlled system increments forward, the optimal trajectory for the states and inputs for the horizon k to k+N is calculated. The states are subject to a upper bound (orange) and the controller makes sure that the state trajectories satesfies these constraints.

2.2 Mathematical optimization

This section serves as a short introduction to mathematical optimization, and the optimization algorithms used in the model predictive controller. Mathematical optimization is an integral part of model predictive control and makes up the majority of the computational burden of the online controller.

The structure and notation of the optimization problems derived in this thesis are defined as in 2.1. with decision variable z, objective function $f(z)$, inequality constraint function $g(z)$, equality constraint function $h(z)$ and lower and upper bounds on decision variable lb and ub. The involved functions can either be linear, quadratic or nonlinear depending on the problem formulation. The optimization problem defined in this thesis consist only of nonlinear functions and parametric and constant constraints. The decision variables that minimize the objective function are refereed to as z^*.

$$\begin{aligned} &\text{Find} && z \\ &\text{to minimize} && f(z) \\ &\text{subject to} && g(z) \leq 0 \\ & && h(z) = 0 \\ & && lb \leq z \leq ub \end{aligned} \tag{2.1}$$

There exist several optimization algorithms that are suitable for solving the nonlin-

ear optimal control problem defined in this thesis. The solvers are selected based on their speed, accuracy and robustness. The solvers examined are two sequential programs, linear and quadratic, these solvers are derived from [6], and two nonlinear solvers interior point line search filter method (IPOPT) [23] and the software company Forces Pros non linear solver, (Forces NLP) [19].

2.2.1 Sequential programs (SQP, SLP)

Sequential programs are iterative methods used to solve nonlinear optimization problems. The programs consist of three main parts. The first part derives a linear approximation of the constraints and either a quadratic or linear approximation of the objective function. The approximation is done by first or second order Taylor series. The second part solves the approximated problem with a linear or quadratic solver. In part three the optimal solution to the approximated problem is used to update the previous solution with step length α. The algorithm for the sequential linear program (SLP) is shown in 1, and sequential quadratic program (SQP) in 2. The variable $z_{V_x}(i)$ is optimal longitudinal speed trajectory derived in sequential itteration i.

Algorithm 1 Sequential linear program (SLP)

$z_i \leftarrow Initial\ guess, i = 1$ ▷ External initial guess
while $RMS(z_{V_x,i+1} - z_{V_x,i}) < \epsilon$ **do** ▷ Convergence criteria
  $\bar{f} = \nabla f(z_i, z_{i+1})$ ▷ Linearized objective function
  $\bar{g} = g(z_i) + \nabla g(z_i, z_{i+1})$ ▷ Linearized inequality constraints
  $\bar{h} = h(z_i) + \nabla h(z_i, z_{i+1})$ ▷ Linearized equality constraints
  $z_{i+1}^{+} = LinearProgram(\bar{f}, \bar{g}, \bar{h}, lb, ub)$ ▷ Solve with suitable solver
  $z_{i+1} = z_i + \alpha\,(z_{i+1}^{+} - z_i)$ ▷ Update optimal solution with step length α
end while

Algorithm 2 Sequential quadratic program (SQP)

$z_i \leftarrow Initial\ guess, i = 1$ ▷ External initial guess
while $RMS(z_{V_x,i+1} - z_{V_x,i}) < \epsilon$ **do** ▷ Convergence criteria
  $\bar{f} = \frac{1}{2}\nabla^2 f(z_i, z_{i+1}) + \nabla f(z_i, z_{i+1})$ ▷ Quadratic objective function
  $\bar{g} = g(z_i) + \nabla g(z_i, z_{i+1})$ ▷ Linearized inequality constraints
  $\bar{h} = h(z_i) + \nabla h(z_i, z_{i+1})$ ▷ Linearized equality constraints
  $z_{i+1}^{+} = QuadraticProgram(\bar{f}, \bar{g}, \bar{h}, lb, ub)$ ▷ Solve with suitable solver
  $z_{i+1} = z_i + \alpha\,(z_{i+1}^{+} - z_i)$ ▷ Update optimal solution with step length α
end while

The convergence criterion of both algorithms is the root mean square (RMS) error of the speed trajectory between the two most recent internal solutions. This criterion is identical to that in [6], where it is determined to be optimal for an MPC energy management application. The RMS convergence limit ϵ and step length α are introduced as tuning parameters that affect the convergence rate of the two programs.

2.2.2 IPOPT

IPOPT (Interior Point OPTimizer) [22] is a nonlinear optimization algorithm designed to deal with large-scale nonlinear programs. It is an open-source optimization algorithm that uses a interior-point line search filter method. IPOPT allows both objective function and constraints to be nonlinear and non convex as long as all functions are twice continuously differentiable. IPOPT is more accurate then sequential programs since it directly evaluates the nonlinear functions rather then linear or quadratic approximations. IPOPT claims to be computationally efficient compared to other solvers, and uses internal scaling of constraints and objective function. The IPOPT solver also claims to be very robust. With all these advantages IPOPT is a viable solution for solving practical nonlinear optimization problems [23]. Because of its accuracy and reliability, IPOPT is used as a benchmark to compare the optimal solutions produced by the faster sequential solvers.

2.3 Literature study: Optimization toolbox

The MPC controller was developed in MATLAB for two reason, to ensure the compatibility with previous work done at Volvo Trucks and the availability of MATLAB toolboxes for solving and building optimization problems.

Two well-known MATLAB toolboxes for handling optimization problems are YALMIP [17] and CasADi [18], which are contrasted at the beginning of the project. These toolboxes all use non-commercial, open-source optimization modeling languages in common. Both toolboxes efficiently interface with MATLAB functions and operators and are designed to hasten and make the creation of model predictive controllers simple. However, as we could only utilize one of them for our project, we conducted a literature study to identify the advantages and disadvantages. The arguments for choosing CasADi are then presented in the following.

CasADi is a toolbox used for the MPC development, it is a free, open-source software package, which can be used in Python, MATLAB, or Octave interfaces, as well as standalone C++ code. It contains a computer algebra system (CAS) for algorithmic differentiation (AD) and a variety of solvers for nonlinear optimal control problems (NOCPs). In addition, it offers a computationally efficient symbolic framework that allows users to construct flexible functions and equations. The flexibility of CasADi is crucial for the development, due to the number of tuning parameters based on real world vehicle testing. In addition CasADi's symbolic framework claims to be highly efficient in constructing Jacobians and Hessians of symbolic functions, which is crucial for efficient sequential programs. Instead, YALMIP is frequently used to address optimization challenges such as linear programming and integer programming. It enables the users to call a variety of external and internal solvers, including cplex, gurobi, and others. The functionality of YALMIP that can be utilized in our project is limited, and external solvers such as Gurobi are not accessible on the internal network of Volvo Trucks due to licensing. We choose not to utilize it in our thesis for this reason.

2.4 Matlab toolbox: CasADi

The controller development for this thesis is carried out in MATLAB using the CasADi toolbox to handle optimization problems, in order to maintain the continuity of the work done by Volvo. The primary component of CasADi, the symbolic framework, enables the users to define and apply expressions such that differentiable functions can be constructed efficiently. The efficient nature of this method stems from the fact that following expressions for derivatives can be generated by applying algorithmic differentiation to the preconceived expressions. In particular, CasADi is flexible for problems constrained by differential equations, which will be addressed in this section.

Before utilizing CasADi, it must be installed and then imported into Matlab's editor with a simple command. For numerical optimum control, CasADi offers a framework of general-purpose building blocks that effectively reduces the amount of effort required to generate a large number of algorithms. To get the more thorough and contemporary introduction to CasADi that is recommended to comprehend detailed example scripts, it is necessary to read the guide [20]. Here, we will skip over the most fundamental operation commands and only present a few specific examples pertinent to the thesis. The showcases presented in this session demonstrate the CasADi syntax and its operation in further detail.

2.4.1 Numerical integration with CasADi

In this thesis, real-world optimal control problems feature a variety of quantity constraints, such as implicitly defined constraints and initial value problems. *Function* objects are introduced to be defined with CasADi symbolic expressions, which supporting numerical evaluation, symbolical evaluation and derivative calculations. For example, with the 'MX' type of CasADi, a RK4 (Runge-Kutta four) algorithm structure can be constructed with around eight lines of code, more over, all the derivatives for any order can be automatically calculated. By employing function *'integrator'* in CasADi, it is possible to embed initial value problem solvers in ODEs or DAEs, as well as calculate all the derivatives of any order. Consider the example of a Van der Pol oscillator, which can be regarded as an ODE with initial value problem:

$$\begin{cases} \dot{x_1} = (1 - x_2^2)x_1 - x_2 + \omega, & x_1(0) = a \\ \dot{x_2} = x_1, & x_2(0) = b \end{cases} \tag{2.2}$$

Where, a and b are constants. Assuming $\omega = 0.5, a = 0, b = 1$, this problem can be addressed with CasADi by using IDAS/CVODES interfaces from the SUNDIALS suite in code 3.

This ODE problem contains initial and terminal constraints, and the CasADi so-

lution is a symbolic representation of the problem, which provide and evaluate a solver instance in order to obtain the result.

Algorithm 3 Example: Integration with CasADi

```
% Import CasADi into Matlab
import CasADi.*

% Construct ODE
(Van der Pol oscillator)
x = SX.sym('x',2);
ω = SX.sym('ω');
z = 1-x(2)^2;
f = [z*x(1)-x(2)+ω; x(1)];
dae = struct('x',x,'ω',ω,...
'ode',f);

% Build integrator
opti = struct('t0',0,'tf',1);
F = integrator('F',...
'vodes',dae,opti);

% Integration
res = F('x0',[0,1],'ω',0.5);
disp(res.xf)
```

2.4.2 Optimization with CasADi

This session will illustrate how to use CasADi synthesis to handle optimal solutions for nonlinear programming. The 'Opti stack' syntax will be introduced for modeling in this part, whereas other NLP solvers can function without it. Considering the NLP defining in 2.3:

$$\begin{aligned} \underset{x,y}{\text{minimize}} \quad & (1-x)^2 + 0.5(y-x^2)^2 \\ \text{subject to} \quad & 1 \leq (x+0.2)^2 + y^2 \leq 4 \end{aligned} \tag{2.3}$$

In contrast to CasADi, code 4 could be employed to model this optimization problem utilizing the 'Opti' stack.

In this example, we select the highly robust nonlinear optimizer IPOPT (Interior Point OPTimizer) as the NLP solver; it finds solutions despite incorrect initial assumptions. Along with IPOPT, our thesis implements other two solvers, SLP (Sequential linear program) and SQP (Sequential quadratic program). The three NLP solvers will be covered in further detail in 3.3.

Algorithm 4 Example: Optimization with CasADi

```
% Import CasADi into Matlab
import  CasADi.*

% Choose Opti stack
opti = CasADi.Opti();

x = opti.variable();
y = opti.variable();

% Modelling the NLP
opti.minimize((1-x)^2+0.5*(y-x^2)^2);
opti.subject_to(1<=((x+0.2)^2 +y^2)<=4);

% Choose solver
opti.solver('IPOPT');

% Solution
sol = opti.solve();
sol.value(x)
sol.value(y)
```

3
Methods

3.1 Space time conversion

In classic mechanics the most common practice is to derive dynamic models in the time domain, but the vehicle model used in the OCP requires road elevation and curvature as input. These road-related properties are most convenient to handle as functions of position, (s) along the vehicles planned route. Building a OCP with a time domain dynamical model that uses space dependant environmental inputs requires additional position estimation potentially resulting in estimation errors [12][13]. This inherent inaccuracy diminishes the trustworthiness of the reference trajectory produced by the MPC, and consequently the potential energy savings and lateral stability in the controlled vehicle [14].

In order to handle this inaccuracy, the dynamical model and OCP constraints are formulated in the space domain. This is done with the method formulated in [6]. Here illustrated in equation 3.1. Where v_x is the longitudinal velocity of the vehicle, (.) represent any differentiable variable. in 3.2 the relationship between longitudinal speed in space and time domain is illustrated.

$$\frac{d(\cdot)}{dt} = \frac{d(\cdot)}{ds}\frac{d(s)}{dt} \approx v_x \frac{d(\cdot)}{ds} \tag{3.1}$$

$$\frac{dv(s)}{ds} = \frac{\dot{v}(t)}{v_x(t)} \tag{3.2}$$

3.2 Optimal control problem

The fundamental energy management control problem for the MPC design is based on the results from [7] and [6] where it is formulated as a nonlinear mixed-integer optimization program (MINLP). The problem can be extended to account for lateral stability as shown in [7] and further explained in 3.2.4, but at its core it consist of an objective function f_{stage}, 3 equality constraints h_{stage}, 12 inequality constraints g_{stage}, and 7 lower and upper bounds on decision variables lb_{stage} and ub_{stage}. With these equations and bounds the full horizon optimal control problem can be built stage wise, using the methods described in 3.3. The one stage optimal control problem is formulated as 3.3.

$$\begin{aligned} &\text{Find} && z = [x, u, u_d]^T \\ &\text{to minimize} && f_{stage}(z) \\ &\text{subject to} && g_{stage}(z) \leq 0 \\ & && h_{stage}(z) = 0 \\ & && lb_{stage} \leq z \leq ub_{stage} \end{aligned} \tag{3.3}$$

The decision variables z are divided into states $x = [v_x, soc, t]^T$, continuous control inputs $u = [F_e, F_{mw}, F_{br}, F_{del}]^T$, Integer control inputs $u_d = \gamma_e$. The involved variables are explained in 3.2.1. The problem is also dependant on several environmental parameters that are functions of road position, consequently written as $param(s)$.

3.2.1 States, inputs and parameters

Both states x and inputs u are decision variables in the OCP. the states are vehicle longitudinal velocity $v_x(s)$, battery state of charge $soc(s)$ and travel time $t(s)$. The continuous inputs are $F_e(s)$ force produced by ICE at its output, $F_{mw}(s)$ wheel force produced by the electric motor, $F_{br}(s)$ brake force from friction brake and retarders other than electric motors. The forth input $F_{del}(s)$ represents the equivalent power dissipation force from the electric drive train. $F_{del}(s)$ is a dummy input and is not passed on as an actuation request but needs to be a decision variable to handle the sign change between retarding and propelling with the electric drive train, as is shown in [6]. All decision variables are gathered in table 3.1. The integer inputs are EM and ICE transmission gears, γ_m and γ_e.

The three space dependant parameters are are road legal speed limit, road grade and road curvature. α is the road grade or slope and is the key parameter for energy management since it dictates how the potential energy of the vehicle change along the planned route. δ is the road curvature and is crucial for calculating the speed limit that guarantees lateral stability. All space dependant parameter are gathered in Table 3.2

All the non space dependant parameters are either environmental or connected to the vehicle specifications. These parameters are further explained in Table 3.2, together with intermediate variables used in the OCP constraint and objective functions.

Table 3.1: Decision variables.

Notation	Definition
v_x	vehicle longitudinal velocity
soc	battery state of charge
t	travel time
F_e	wheel force form ICE
F_{mw}	wheel force form EM
F_{br}	wheel force form friction brakes
F_{del}	equivalent dissipated force

Table 3.2: Variables and parameters.

Notation	Definition
F_f	equivalent engine fuel force
F_{mc}	equivalent force between the battery and EM
F_{db}	equivalent force dissipated in the battery
F_{ew}	equivalent force at the ICE transmission output
F_a	auxiliary force
F_{dm}	equivalent forces dissipated in the EM
F_{dmt}	equivalent forces dissipated in the EM transmission
p_{bmin}	the minimum battery power
p_{bmax}	the maximum battery power
t_{ref}	reference trip time
γ_e	ICE gear
γ_m	EM gear
m	vehicle total mass
g	gravitational acceleration
α(s)	road grade
δ(s)	road curvature
s	distance traveled
f_r	rolling resistance coefficient
ρ_a	air density
A_f	equivalent vehicle front area
c_d	air drag coefficient
R_w	wheel radius
r_e	gear ratio from the wheel to ICE
T_e	ICE torque
ω_e	ICE speed
r_m	gear ratio from the wheel to EM
T_m	EM torque
ω_m	EM speed
$a_{ij}, b^m_{ij}, b^e_{ij}, h^{+,-}_{ij}$	the coefficients of the fitted functions
V_b	battery voltage
R	battery resistance

3.2.2 Energy management

The vehicle models investigated are equipped with a parallel hybrid powertrain. The powertrain is illustrated in figure 3.1 taken from [6] and demonstrates the energy flow balance (input/output power P) between the multiple powertrain subsystems in a hybrid vehicle. There are two types of arrows depicted in this figure. The blue arrow represents the direction of energy flow, whereas the red arrow illustrates that energy can also flow in one of the other directions. In the fuel tank and battery components, a dot before the arrow indicates that they are power sources.

The energy flow balance assumes that no energy is stored in any subsystem except the fuel tank and battery, therefore all inertia of rotating components are neglected, and it does not account for potential energy losses due to slippage between the ground and the tyre. The energy flow balance is the basis of the objective function in the OCP. The forces defined in 3.2.1 are all derived by time integration of the powers described in the energy flow balance, as is indicated by the notation in table 3.2. An further explanation off the individual energy flows are shown in table 3.3.

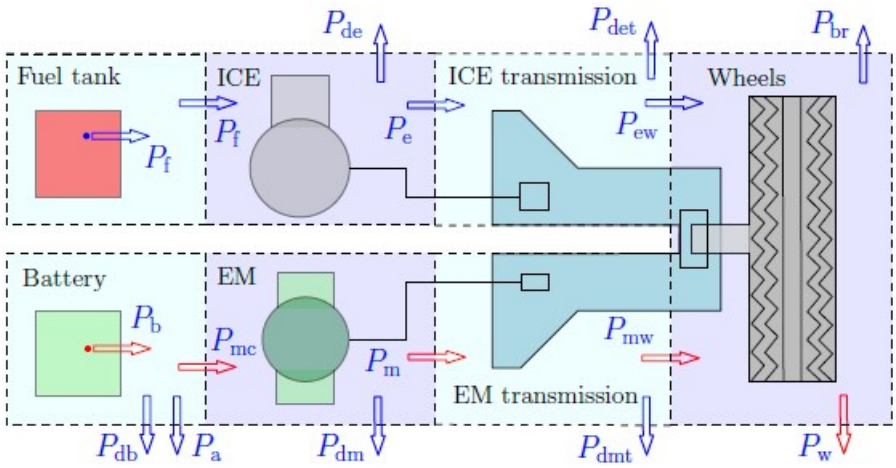

Figure 3.1: Energy flow balance in the hybrid powertrain subsystems with arrows illustrating the direction of the energy flow. This is the same system description and figure that is used in [6].

Table 3.3: Variables in the the energy flow balance in the hybrid powertrain system.

Notation	Definition
P_f	fuel power
P_e	power at the ICE output
P_{de}	power dissipated in the ICE
P_{det}	power dissipated in the transmission between ICE and wheels
P_{ew}	power at the ICE transmission output
P_{br}	power for the friction brake (or engine brake power)

Notation	Definition
P_b	power provided by or stored in the battery
P_{db}	power dissipated in the battery
P_a	power used for the auxiliaries
P_{mc}	power consumed or regenerated by the electric motor
P_m	power at the EM output to or at the input from the transmission
P_{dm}	power dissipated in the EM
P_{dmt}	power dissipated in the EM transmission
P_{mw}	EM power at the output to or the power at the input from wheels
P_w	power at wheels

3.2.2.1 Objective function

The objective function for one stage f_{stage}, shown in equation 3.4 is based on the energy flow balance. The goal of minimizing the objective function is to minimize the fuel consumption, the electric drive train energy dissipation and the use of friction brakes while maximizing the use of the electric motor for propulsion. The equivalent fuel force is calculated with equation 3.5, which is a polynomial fitted to measured fuel consumption data, with coefficients a_{ij}. The approximation is based on the results from [6]. In [6] it is shown that depending on the degree of the fitted polynomial the resulting objective function can be both quadratic and convex. This particular polynomial degree that is quadratic and has a positive semidefinite hessian is used for the objective function.

$$f_{stage}(x, u, u_d) = F_f(x, u, u_d) + F_{br} - F_{mw} + F_{del} \tag{3.4}$$

$$F_f(x, u, u_d) = \frac{1}{v_x} \sum_{i=0}^{5} \sum_{j=0}^{5} a_{ij} (\frac{r_e(\gamma_e)}{R_w} v_x)^i (\frac{R_w}{r_e(\gamma_e)} F_e)^j \tag{3.5}$$

3.2.2.2 Equality constraints

The equality constraints for one stage h_{stage} consist of the dynamic model equations. These equations define how the state trajectories are affected by the inputs, and in the discrete world define how the states progress from one stage to the next. In addition to the three model equations the initial stage in a control horizon has an additional equality constraint to make sure that the controller initializes the problem at the states measured from the plant. The definition of the one stage equality constraints are shown in 3.6.

$$h_{stage}(x, u) = [h_{v_x}, h_{soc}, h_t]^T, first\ stage:\ h_{stage}(x, u) = [h_{x0}, h_{v_x}, h_{soc}, h_t]^T \tag{3.6}$$

The dynamic vehicle model is a lumped mass model that describe longitudinal acceleration along the planned route and is the same model used in [6], and shown in equation 3.7, the road grade α is defined as positive down hill. Battery state of charge dynamics are defined in 3.8, due to the fact that the elctric motor can both propel and retard the veichle with there are two dynamic equations for each case,

F_{mc}^{+} and F_{mc}^{-} shown in equations 3.11 and 3.12. The equations are taken from [6] where they are derived as polynomials with coeficients $h_{i,j}^{+,-}$ fitted to experimental data. The time dynamics are defined in equation 3.9. The initial state equality is defined in equation 3.10.

Vehicle longitudinal dynamics:

$$h_{v_x} = \frac{dv_x}{ds} - \frac{1}{mv_x}(-F_{br} + \eta_{te}F_e + F_{mw} + mg\sin\alpha - mgf_r\cos\alpha - 0.5\rho_a A_f c_d v_x^2) = 0 \tag{3.7}$$

Battery SOC dynamics:

$$h_{soc} = \frac{dsoc}{ds} - \frac{1}{E_{\text{bmax}}}(F_{del} + F_{mw} + F_a) = 0 \tag{3.8}$$

Time dynamics:

$$h_t = \frac{dt}{ds} - \frac{1}{v_x} = 0 \tag{3.9}$$

initial stage constraint:

$$h_{x0} = [v(s_0), soc(s_0), t(s_0)]^T - [v_0, soc_0, t_0]^T = 0 \tag{3.10}$$

Electric motor propulsion:

$$F_{mc}^{+} = \sum_{i=0}^{5}\sum_{j=0}^{5} h_{ij}^{+}\left(\left(\frac{r_m(\gamma_m)}{R_w}\, v_x(s)\right)^i\left(\frac{R_w\, F_{mw}}{r_m(\gamma_m)\eta_{tm}}\right)^j\right) \tag{3.11}$$

Electric motor retardation:

$$F_{mc}^{-} = \sum_{i=0}^{5}\sum_{j=0}^{5} h_{ij}^{-}\left(\left(\frac{r_m(\gamma_m)}{R_w}\, v_x(s)\right)^i\left(\frac{R_w\, F_{mw}\, \eta_{tm}}{r_m(\gamma_m)}\right)^j\right) \tag{3.12}$$

3.2.2.3 Inequality constraints

The one stage inequality constraints g_{stage} related to physical limitations are torque limits for the ICE and EM, and limits on battery charge and discharge power. In addition to these physical constraints the electric drive train dissipation force F_{del} is approximated as a region lying in between two inequalities. In total there are twelve inequality constraints present for each stage.

The constraints related to torque limits for the ICE are shown in equation 3.13. The torque limits are based on results from [6] where they are derived as 4 polynomials with degree j with coefficients $b_{i,j}^{e}$ that is fitted to measured experimental data. Each of the fitted polynomials act as individual inequality constraints which means that the dimension of g_{F_e} is $[4x1]$. The EM torque limits are also introduced as polynomials derived from experimental data, with coefficients $b_{i,j}^{m}$. There are 2 polynomials of degree j that define the upper and lower limits. Here shown in equation 3.14 and 3.15. Each of the polynomials are handled separately which gives

$[g_{F_{mw,max}}, g_{F_{mw,min}}]^T$ dimension $[4 \times 1]$. The battery charge and discharge power constraints are derived from [6] and are define as $g_{pb,min}$ and $g_{pb,max}$ in equations 3.17 and 3.16.

Due to the fact that the EM can both propel and retard the vehicle the forces related to the electric drive train can either be positive or negative, the model equations that describe the dissipation force in the electric drive train for these two cases, is a piece-wise function with no derivative in the origin. The derivative discontinuity is handled by splitting up the equality constraint into two inequality constraints here shown in equation 3.18 and 3.19. By including the F_{del} term in the cost function, when minimizing, F_{del} is squeezed in between these two constraints and pushed towards the boundaries, i.e., an equality constraint is satisfied.

ICE torque upper limit:

$$g_{Fe} = \frac{R_w}{r_e(\gamma_e)} F_e - \min\{\sum_{j=0}^{3} b_{ij}^e (\frac{r_e(\gamma_e)}{R_w} v_x)^j\} \leq 0, i = 1, \ldots, 4 \tag{3.13}$$

EM torque limits:

$$g_{F_{mw,max}} = \frac{R_w F_{mw}}{r_m(\gamma_m)\eta_{tm}} - \min\{\sum_{j=0}^{3} b_{ij}^m (\frac{r_m(\gamma_m)}{R_w} v_x)^j\} \leq 0, \; i = 1, 2 \tag{3.14}$$

$$g_{F_{mw,min}} = -\frac{R_w F_{mw} \eta_{tm}}{r_m(\gamma_m)} + \max\{\sum_{j=0}^{3} b_{ij}^m (\frac{r_m(\gamma_m)}{R_w} v_x)^j\} \leq 0, \; i = 3, 4 \tag{3.15}$$

Battery power limits:

$$g_{pb,min} = p_{\text{bmin}} - v_x(F_{del} + F_{mw} + F_a) \leq 0 \tag{3.16}$$

$$g_{pb,max} = v_x(F_{del} + F_{mw} + F_a) - p_{\text{bmax}} \leq 0 \tag{3.17}$$

$$g_{F_{del,+}} = F_{del} - F_{mc}^{+} - F_{mw} + \frac{R v_x}{V_b^2}(F_{mc}^{+} + F_a)^2 \leq 0 \tag{3.18}$$

$$g_{F_{del,-}} = F_{del} - F_{mc}^{-} - F_{mw} + \frac{R v_x}{V_b^2}(F_{mc}^{-} + F_a)^2 \leq 0 \tag{3.19}$$

3.2.2.4 Bounds on states and inputs

In addition to the equation based inequality constraints there are direct constraints or bounds on the states and inputs of the NOCP. These bound serve two purposes, to speed up the optimization algorithm by shrinking the problems feasible space, and to describe desired behaviour of the controlled system. There are one upper and lower bound for each decision variable per stage.

The bounds that shrink the feasible space are lower bound on on longitudinal speed lb_{v_x} shown in equation 3.20, which is introduced as a fraction κ in range $(50 - 80)\%$ of the reference speed generated as a initial guess v_{ref}, further explained in 3.2.4.2. v_{ref} is space dependant and can change for each stage in the horizon. Upper and

lower bounds on time ub_t and lb_t, shown in equation 3.21 where the lower bound is based on the measured time from the plant at the first stage of the horizon, and the upper bound is the final time generated from the initial guess. Upper and lower bounds on all input forces shown in equation 3.22, defined with a constant F_{max} in range $(10^5 - 10^6)$.

The bounds that define the desired behavior of the controlled system are upper and lower bounds *soc*, shown in equation 3.23 these serve to protect the battery and extend its lifetime, the parameters $(soc_{min} - soc_{max})$ are usually in the range $(20 - 80\%)$. A very important state constraint is the upper bound on longitudinal speed v_{max}, this constraint is used to guarantee lateral stability of the vehicle. v_{max} can either be generated by the methods introduced in 3.2.4.2, or derived with lateral dynamics as described in 3.2.4. With both methods the limit v_{max} is space dependant and can change for each stage in the horizon.

There is an additional constraint on battery state of charge *soc* at either the end of the horizon or end of the planned route. This constraint can either be used to make sure that there is as much charge at the start and end of each horizon, or at the end of the planned route. Without this constraint the energy optimal solution is to drain the battery down to soc_{min} since there is no penalty in the cost function regarding *soc*.

Bound on on longitudinal speed:

$$\frac{v_{ref}}{\kappa} \leq v_x \leq v_{max} \tag{3.20}$$

Bound on time:

$$t_{ref,0} \leq t \leq t_{ref,N}, \tag{3.21}$$

Bound on Force inputs:

$$[0, -F_{max}, 0, 0]^T \leq [F_e, F_{mw}, F_{br}, F_{del}]^T \leq F_{max} \tag{3.22}$$

Bounds on battery SOC:

$$soc_{min} \leq soc \leq soc_{max} \tag{3.23}$$

$$\tag{3.24}$$

3.2.3 Gear optimization

To simplify the NOCP the strategy for gear optimization derived in [6] and [7] is used. The strategy consist of removing the integer states representing the ICE and EM gear choice γ_e and γ_m from the optimization problem from section 3.2.2, and instead treat them as stage dependant parameters. The optimal gears are instead found with a simple one step optimization algorithm defined in 5. The problem is further simplified by excluding electric motor transmission gears completely and only trying to find the energy optimal ICE gears. It is important to stress that the algorithm used does not guarantee either local or global optimality of the MINLP.

The gear optimization problem can be expressed as in 3.25. Where the decision variable is the ICE gear for each stage in the horizon, $\boldsymbol{\gamma_e}$. The objective function is the equivalent fuel force F_f. The constraints are the ICE torque limits, $\mathbf{g_{F_e}}$ defined in equation 3.13 and ICE rpm limits which can be expressed as a function of longitudinal speed v_x and gear ratio, shown in equation 3.26. Here all functions and variables in bold represent all stages in the horizon, i.e. they are vectors or vector functions of length N. The construction of the gear optimization problem is further explained in section 3.3.1.

$$\begin{aligned} &\text{Find} && \boldsymbol{\gamma_e} \\ &\text{to minimize} && \mathbf{F_f}(x, u, \gamma_e) \\ &\text{subject to} && \mathbf{g_{F_e}}(x, u, \gamma_e) \leq 0 \\ & && \boldsymbol{\omega}(x, \gamma_e) - \omega_{max} \leq 0 \\ & && \omega_{min} - \boldsymbol{\omega}(x, \gamma_e) \leq 0 \end{aligned} \tag{3.25}$$

$$\boldsymbol{\omega}(v_x, \gamma_e) = \frac{\mathbf{v_x}}{R_w} r_e(\gamma_e) \tag{3.26}$$

The gear optimization algorithm 5 for the ICE gears γ_e is a function that takes the optimal solution to the problem defined in 3.2.2, x^* and u^* and the gear sequence of length N used for that solution, $\bar{\gamma_e}$ as inputs, and as an output it gives a more energy efficient gear sequence for the given state and input trajectories γ_e^*. The possible gear choices for the Volvo trucks used as models are $K = [1 - 12]$. The gear optimization is based on first finding all the infeasible gears for each stage in a horizon, for any infeasible gears a penalty is added to the penalty matrices $V_{g_{F_e}}$ and V_ω. The penalty is much larger then any value in the range of F_f which makes sure that the minimum can never be a infeasible gear. The optimal gear is found by finding the column index of the minimum value of M_{F_f} in row j with penalties added. Where M_{F_f} is a matrix containing the equivalent fuel force for all possible gears K in the horizon N. This can be done with a built-in function in most high level programming languages. To avoid changing gears to0 frequently the decrease in F_f for the new gear γ_e^* has to be lower then some fraction ϵ. There are additional features to ensure the robustness of the algorithm, if no gears are feasible the gears will not be updated, and if the ICE engine is turned off the gears are chosen to be in the middle of the feasible RPM range.

Algorithm 5 ICE gear optimization

$V_{g_{F_e}} = \mathbf{0}$ ▷ Penalty matrix with dimension $[N, K]$
$V_{\omega} = \mathbf{0}$ ▷ Penalty matrix with dimension $[N, K]$
for $i \leftarrow 1$ to $K, \gamma_e \in K$ **do** ▷ Evaluate problem 3.25 for all possible ICE gears
  $M_{F_f} = \mathbf{F_f}(x^*, u^*, i)$ ▷ Build matrix with $\mathbf{F_f}$ values, dimension $[N, K]$
  $M_{g_{F_e}} = \mathbf{g_{F_e}}(x^*, u^*, i)$ ▷ Build matrix with $\mathbf{g_{F_e}}$ values, dimension $[N, K]$
  $M_{\omega} = \boldsymbol{\omega}(x^*, i)$ ▷ Build matrix with $\boldsymbol{\omega}$ values, dimension $[N, K]$
  for $j \leftarrow 1$ to N **do** ▷ Check each stage in horizon N
    if $M_{g_{F_e},[j,i]} \geq 0$ OR $M_{\omega,[j,i]} \geq \omega_{max}$ OR $M_{\omega,[j,i]} \leq \omega_{min}$ **then** ▷ Find infeasible gears for stage j
      $V_{g_{F_e},j,i} = P$ ▷ Add penalty P to infeasible gears i in stage j
      $V_{\omega,j,i} = P$ ▷ Add penalty P to infeasible gears i in stage j
    end if
    $\gamma^*_{e,j} = \min(M_{F_f,j} + V_{g_{F_e},j} + V_{\omega,j})$, w.r.t i ▷ Find γ^*_e for stage j
    if $\frac{F_f(x^*,u^*,\gamma^*_e)}{F_f(x^*,u^*,\bar{\gamma_e})} < \epsilon$ **then** ▷ Evaluate fuel savings with new gears
      $\gamma^+_e = \gamma^*_e$ ▷ Update gears for stage j
    end if
  end for
end for

3.2.4 Safety: lateral stability

A vehicle is considered articulated if it has a permanent or semi-permanent pivot joint that enables it to turn more abruptly. Any vehicle towing trailers (dollies), including the semi-trailer, would be referred to as articulated in a more general manner. With distributed propulsion and braking as well as a multi-unit long vehicle structure, the articulated vehicle under consideration in this thesis possesses a construction that makes it susceptible to losing yaw stability. Trailer swing and jackknifing are among the most typical precarious circumstances where lateral stability is compromised. Trailer swing, also widely recognized as trailer slew, is the term for when a trailer of an articulated vehicle slides to one side. It frequently takes places on a wet, slick road surface or in tight turns and cants, while the driver applied the brakes. The process of collapsing an articulated vehicle into a configuration that is similar to an acute angle is referred as jackknifing. It derives the name from the folding pocket knife, which is also illustrative of its understandability. In the event that a trailer is being towed by a vehicle, the trailer will push the towed vehicle from behind and force it to turn around when the vehicle starts to skid, resulting in a collision between the vehicle units [27]. When the lateral instability situation is getting worse, which is a very dangerous situation that can cause the vehicle to be damaged as the driver loses control, jackknifing appears to occur.

Unlike jackknifing, trailer swing (trailer slew) does not cause as significant damage to the vehicle since the trailer will realign as the vehicle keeps traveling forward after the driver releases the brakes [27]. The issue is that, in order to save energy, it is preferred to use the electrical motor as much as possible. However, the worst case

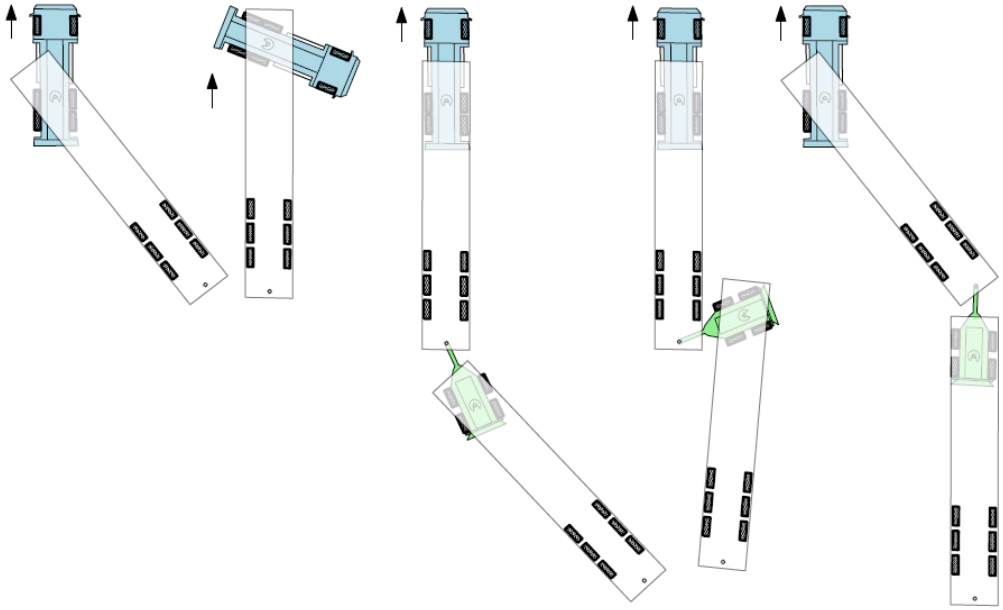

Figure 3.2: Trailer swing and jackknifing in articulated vehicles. [7]

jackknifing happens when the energy management controller estimates that it will be less energy cost to only uses the electric drive axle for braking and propulsion, which means the propulsion and braking are distributed incorrectly. The figure 3.2 provides an illustration of what trailer swing and jackknifing in trucks and articulated vehicles look like.

There are several approaches to achieving the goal of ensuring lateral stability, one of which is to extend the vehicle dynamic model to include the lateral dynamics. Another alternative is to set speed limits with a rule based approach. However, the drawback of the latter is that it is derived only based on the lateral acceleration and thus it neglects the effect of combined slip as discussed in [7]. Moreover, the energy management is decoupled from the lateral motion control which results in mismatch between the models and consequently an energy loss. Therefore, the first approach is preferable.

3.2.4.1 Including lateral dynamics

The vehicle model that includes lateral dynamics for a A-dobule truck with a electric dolly is taken from [7], and illustrated in figure 3.4. The model equations are generated with the software defined in [8]. The equations are implicit differential equations and are defined on the same form as in equation 3.29, the full equations are available in [7] and [8]. The equations from [8] are converted into the space domain using the method defined in 3.1. The state vector for the lateral dynamics is defined in 3.28, with X and Y being the global position of the first unit, ϕ is the angle between the unit x axis and the global x axis, θ_{1-3} is the angle between the vehicle units.

The lateral dynamics from [8] are augmented to have the same inputs as the longitudinal dynamics from 3.2.2.2, this is done with a mapping matrix A_u which maps the force inputs u to a wheel group force, similar to the strategy defined in [7], the mapping matrix A_u is defined in 3.27. The wheel groups defined for an A double with electric dolly are shown in figure 3.3. The figure is taken from [7].

$$A_u = \begin{bmatrix} 0 & 0 & \frac{-1}{10} & 0 \\ 1 & 0 & \frac{-2}{10} & 0 \\ 0 & 0 & \frac{-3}{10} & 0 \\ 0 & 0 & \frac{-1}{10} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-3}{10} & 0 \end{bmatrix}, u_{lateral} = A_u \begin{bmatrix} F_e \\ F_{mw} \\ F_{br} \\ F_{del} \end{bmatrix} \tag{3.27}$$

$$x = [X, Y, \phi, \theta_1, \theta_2, \theta_3, v_x, v_y, \frac{d\phi}{ds}, \frac{d\theta_1}{ds}, \frac{d\theta_2}{ds}, \frac{d\theta_3}{ds}]^T \tag{3.28}$$

$$F(x, \frac{dx}{ds}, A_u\, u, \alpha(s), \delta(s), param) = 0 \tag{3.29}$$

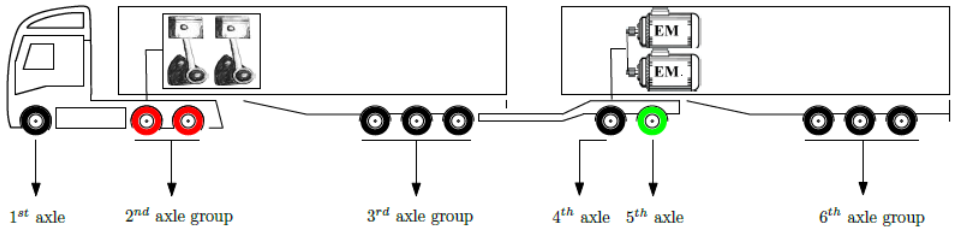

Figure 3.3: Illustration of an A-double with e-dolly and the wheel groups used for the multibody lateral dynamics equations. The figure is taken from [7].

With the lateral dynamics it is possible to directly constrain vehicle unit acceleration and tire forces to assure lateral stability. This is done by adding two additional inequality constraints. The first constraint is related to tire forces and slip, and uses the method from [7] where the friction ellipse model is used to calculate the lateral tire forces F_y, the longitudinal tire forces F_x are simply $u_{lateral}$ from 3.27, and F_z is the vertical forces on each tire group and μ is the road friction. The constraint can be further tuned with a safety factor β. The constraint is shown in equation 3.30.

$$g_{Fy} = \sqrt{(A_u\, u)^2 + F_y(x, A_u\, u)^2} - \frac{1}{\beta}\, \mu\, F_z \leq 0 \tag{3.30}$$

The unit lateral acceleration is similarly constrained, but with respect to a fixed allowed value for acceleration $a_{y,max}$, and is displayed in equation 3.31.

$$g_{ay} = |a_y(x, A_u\, u)| - a_{y,max} \leq 0 \tag{3.31}$$

The lateral dynamics F, and constraint functions $F_y(x, A_u\, u)$ and $a_y(x, A_u\, u)$ are large nonlinear equations that are computationally expensive to evaluate. Because of this a simplified strategy to include the lateral dynamics is introduced in 3.3.5.

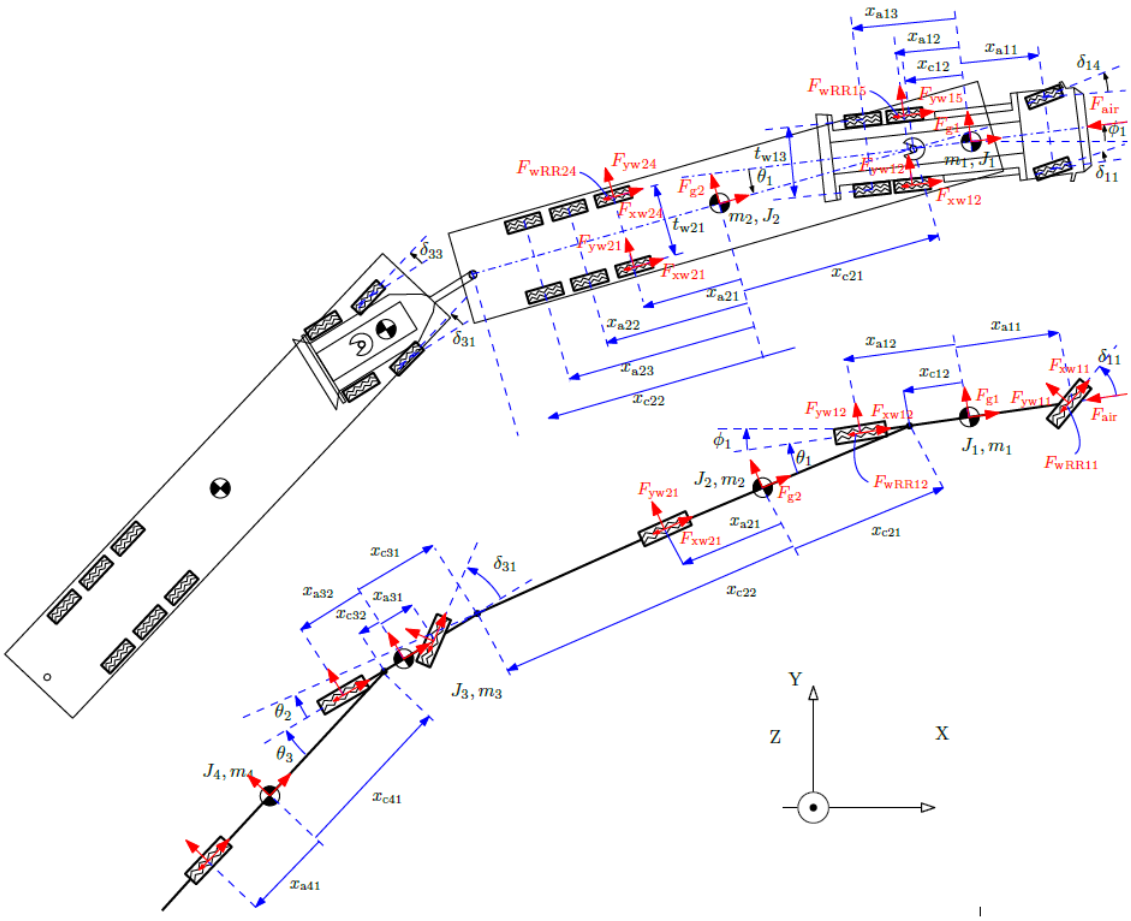

Figure 3.4: Schematic figure of relevant angels and forces for the A-double combination vehicle as a two and one wheel model. The figure is taken from [7].

3.2.4.2 Rule based speed limitation and initial reference trajectory generation

The rule based speed limitation and initial reference trajectory generation is done with Volvo software. The software is based on the method derived in [6] and generates both initial guesses for state and input trajectories, as well as a limitation to longitudinal speed that guarantees lateral stability. The initial guess is crucial to speed up the computation time at startup for all solvers used in this thesis, and it is needed even if lateral dynamics are included in the MPC model.

The initial reference trajectory describe how a diesel driven combination vehicle drives the planned route as fast as possible without exceeding some predetermined limit of lateral acceleration for the center of gravity for the whole combination vehicle. The software produces the following trajectories: the speed profile along the planned route $v_{x,ref}$, the time trajectory t_{ref} and most crucially an arrival time $t_{ref,end}$, feasible ICE gears $\gamma_{e,ref}$, ICE forces used to propel the vehicle $F_{e,ref}$.

The lateral stability speed limitation produced by the software $v_{x,max}$ is a much

simpler method compared to including lateral dynamics in the MPC model, due to the methods simplicity it needs to be more conservative w.r.t. the maximum speed allowed along the planner route. This is not the ideal approach since the conservative speed limitation has a negative effect on energy optimization, as is shown in [7]. It is however a very computational efficient, and does not affect the controllers online computation time, since the limitation can be derived offline. An example of how the rule based lateral stability speed constraint for a 2 km route is displayed in figure 3.5

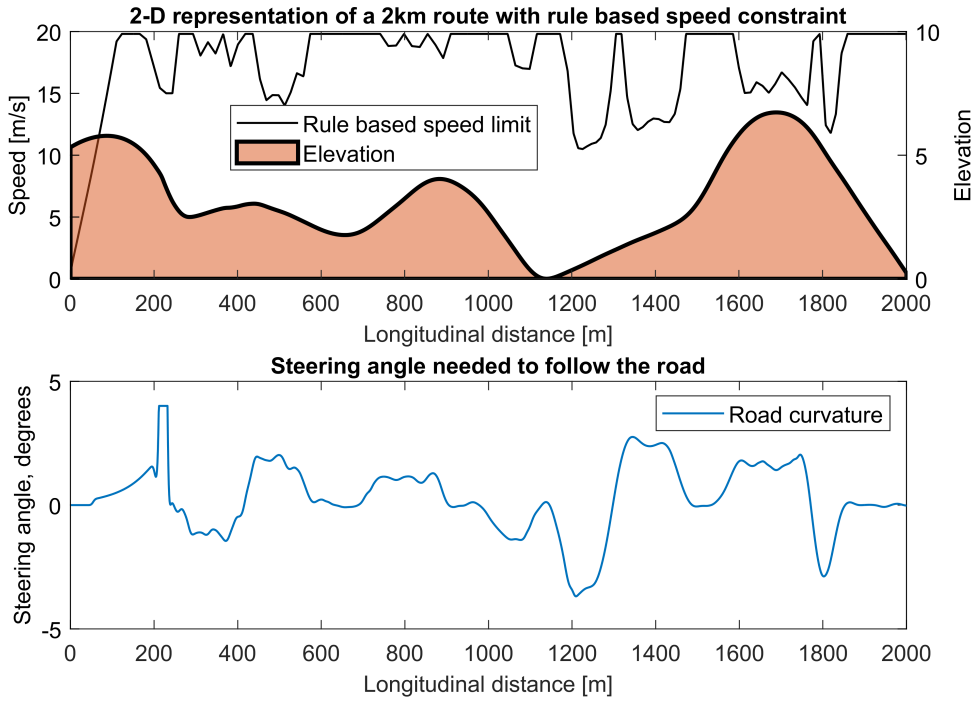

Figure 3.5: Example of the rule based speed constraint and corresponding steering angles for a 2km route.

3.3 MPC with CasADi

In this section the method used for building the optimal control problem and different types of MPCs using the CasADi toolbox is described. The MPC building process consists of 3 offline steps, The first step handles how the NOCP is built using CasADi symbolics and how the sparsity patterns of the constraint jacobians can be changed depending on this process, secondly how discretization of the model dynamics is done, and lastly how space dependant state and input constraints are passed to the online controller. In addition to this the implementation of the different optimization algorithms or solvers are defined for the MPC with the two different methods for lateral stability, together with a method for including a shrinking horizon when the vehicle is closing in on the end of its planned route.

3.3.1 Symbolic NOCP generation with CasADi and sparsity patterns

CasADi supplies two types of symbolic variables, SX and MX. The SX symbolic are best used for mathematical operations in R^1, and the MX on the contrary is more efficient when used for matrix operations. Since none of the equations defined for the NOCP are based on matrix operations SX symbolics is used to the largest extent possible. In general expressions built with SX take up more memory than ones built with MX but since memory space is not a concern this is not considered an issue for this thesis.

The NOCP is built in a Matlab function that take vehicle parameters that are independent of space $param$, control horizon N and discritization stepsize ds as input. The output is the nonlinear constraint functions and objective function of the NOCP and their respective jacobians, all functions are defined for horizon N and discretized with stepsize ds. Both the decision variable vector $\mathbf{z}$ and inequality and equality constraint functions are built stage wise using the equations defined in 3.2.2.3 and 3.2.2.2, but as in CasADi symbolic functions, indicated by sym. This is done to ensure a diagonal and dense sparsity pattern for their respective jacobians. This can be done with either a for loop, or the built-in CasADi function $CasADi.function.map(N)$, which is more efficient to evaluate.

The equality constraint are built using the discretized, denoted by $\bar{}$ dynamics $\bar{h}_{stage}$, defined in 3.3.2. The space dependant parameters $p(s)$ and decision variables $z(s)$ are defined as inputs to the constraint functions while the non space dependant parameters $param$ are defined as constants in the symbolic expression and are not inputs in the final constraint functions. The resulting constraints functions are defined as in 3.32 and 3.33. With inputs in bold indicating that they are vectors for the whole control horizon N.

$$\text{Equality constraints whole horizon:} \tag{3.32}$$
$$\mathbf{h}(\mathbf{z}, \mathbf{p}) = \bar{h}^{sym}_{stage}(z(s), p(s), param).map(N)$$
$$\text{Inequality constraints whole horizon:} \tag{3.33}$$
$$\mathbf{g}(\mathbf{z}, \gamma_\mathbf{e}, \mathbf{p}) = g^{sym}_{stage}(z(s), \gamma_e(s), p(s), param).map(N)$$

The objective function for the whole horizon is built as a sum of the stage wise objective functions, defined in 3.2.2.1. With the same symbolic strategy as for the constraints. The final function is defined as in 3.34

$$\mathbf{f}(\mathbf{z}, \gamma_\mathbf{e}, \mathbf{p}) = \sum_{k0}^{k=N} f^{sym}_{stage}(z(s), \gamma_e(s), p(s), param) \tag{3.34}$$

The resulting vector with decision variables for the whole horizon has the form defined in 3.35, and the constraint functions are defined in 3.36 and 3.37. With each stage being indicated by s_i, it comes naturally from the space dynamic discretization that the stages are defined a distance of ds apart. In addition to building the

constraint and objective function as CasADi symbolics, the fuel equivalent force function $\mathbf{F_f}$, and ICE torque constraint $\mathbf{g_{F_e}}$ are created and passed to the gear optimization problem.

Decision variable vector

$$\mathbf{z} = [x_{s0}, u_{s0}, x_{s1}, u_{s1}, ..., x_{sN}, u_{sN}, x_{sN+1}]^T \tag{3.35}$$

Equality constraint whole horizon

$$\mathbf{h}(\mathbf{z}, \mathbf{p}) = [\bar{h}_{s_0}(\mathbf{z}(s_0), \mathbf{p}(s_0)), ..., \bar{h}_{s_N}(\mathbf{z}(s_N), \mathbf{p}(s_N))]^T \tag{3.36}$$

Inequality constraints whole horizon:

$$\mathbf{g}(\mathbf{z}, \gamma_\mathbf{e}, \mathbf{p}) = [g_{s_0}(\mathbf{z}(s_0), \gamma_\mathbf{e}(s_0), \mathbf{p}(s_0)), ..., g_{s_N}(\mathbf{z}(s_N), \gamma_\mathbf{e}(s_N), \mathbf{p}(s_N))]^T \tag{3.37}$$

The jacobian of $\mathbf{h}$ and $\mathbf{g}$ w.r.t the decision variables $\mathbf{z}$ is derived with the built in matlab function $jacobian()$ that calculates the jacobian of a symbolic expression with respect to a symbolic variable. The resulting symbolic expression is then defined as a CasADi function with the same inputs as the original function, defined as $\mathbf{J_h}(\mathbf{z}, \mathbf{p})$ and $\mathbf{J_g}(\mathbf{z}, \gamma_\mathbf{e}, \mathbf{p})$. The jacobian and hessian of the objective function is derive similarly and defined as $\mathbf{J_f}(\mathbf{z}, \gamma_\mathbf{e}, \mathbf{p})$ and $\mathbf{H_f}(\mathbf{z}, \gamma_\mathbf{e}, \mathbf{p})$. The sparsity pattern of the jacobians and hessian is shown in figure 3.6.

3.3.2 Discretization with CasADi

The continuous space dependant dynamic equations need to be discritized before they can be included in the MPC. Discretization is done with CasADis numerical integration method $Integrator()$ with stepsize ds for horizon N. The integration method used for the NOCP depends on which vehicle dynamic model that is used. For the energy management problem with a rule based lateral stability constraint the dynamic equations are ordinary differential equations (ODE) and can be solved with an explicit Runge-kutta method. For a problem that includes lateral dynamics the model equations are implicit, and are first transformed into a differential algebraic equation (DAE) and solved with the implicit integrator IDAS from the Sundials suite , which is part of the CasADi toolbox.

Both methods of integration follow the same pattern. The dynamic equations are evaluated with variables declared as symbolic expression, the symbolic expressions are then passed to the CasADi Integrator method F where differentiable state, variable parameters, algebraic states and integration step size ds is declared. The integrator method is then called N number of times in a for loop where each call depends on the result of the previous iteration. This method builds a $[N \cdot n, 1]$ symbolic expression which contains a chain with N number of integration steps with n being the number of states in the dynamic equations, the expression is declared as a CasADi function with the same inputs as 3.36 and can directly be evaluated as a nonlinear equality constraint in MPC optimization algorithm. The algorithm is displayed in 6.
The IDAS implicit integration strategy is compared with Matlabs ode15i to ensure that they are stable and that the symbolic functions retain sufficient accuracy, and

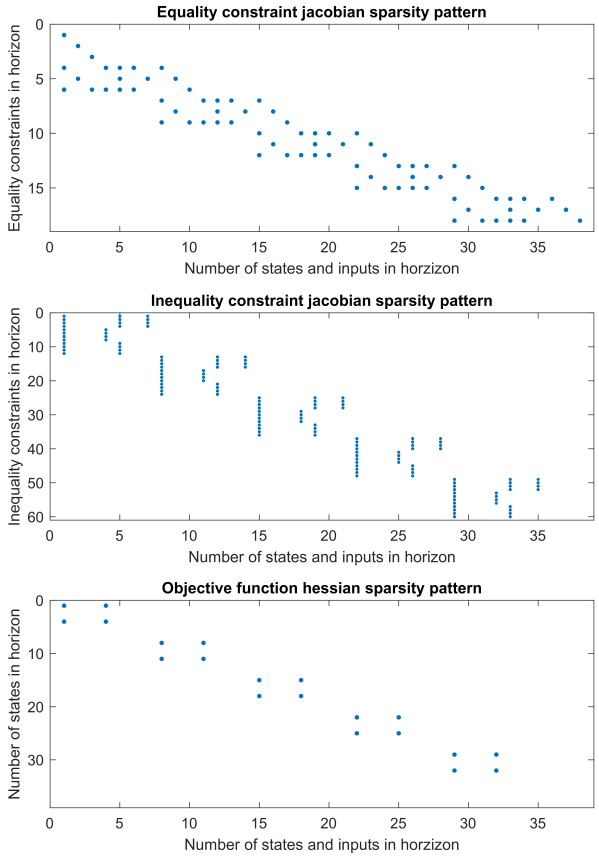

Figure 3.6: Figure ilustrating the sparsity pattern of $\mathbf{J_h}(\mathbf{z}, \mathbf{p})$, $\mathbf{J_g}(\mathbf{z}, \gamma_\mathbf{e}, \mathbf{p})$ and $\mathbf{H_f}(\mathbf{z}, \gamma_\mathbf{e}, \mathbf{p})$ without lateral dynamics and with horizon $N = 75m$ and stepsize $ds = 15m$. Resulting in dimensions: $\mathbf{J_h}(\mathbf{z}, \mathbf{p}) = [18 \times 38]$, $\mathbf{J_g}(\mathbf{z}, \gamma_\mathbf{e}, \mathbf{p}) = [60 \times 38]$, $\mathbf{H_f}(\mathbf{z}, \gamma_\mathbf{e}, \mathbf{p}) = [38 \times 38]$.

stability. The simulation comparison shown in figure 3.7 is done on a flat road with one left and right turn. The step size used in IDAS is 1m with a maximum of 4 intermediate steps, the step size used in ode15i is much smaller. This is determined to be the cause of the discrepancy between the two simulation results.

3.3.3 Updating space dependant constraints and parameters

As mentioned in 3.2.1 and 3.2.2.4 there are both space dependant parameters and constraint limits. The space dependant parameters are generated from road data available at Volvo, and interpolated or extrapolated to fit the discretization step size ds. The reference trajectories calculated with the method defined in 3.2.4.2 are also discretized with the step size ds. The resulting data files are then passed to the MPC controller which can read them from its current longitudinal position s and $s + ds \cdot N$ meters in the future. Both data files can involve roads longer or sorter

Algorithm 6 Symbolic integration with CasADi

$F_{s->s+ds}(x, u, p) = h^{sym}_{stage}(x, u, p)$ ▷ Symbolic integration step with stepsize ds, and symbolic ODE or DAE from equation 3.6 or 3.29.
for $i \leftarrow 1$ to N **do** ▷ Build N number of integration steps
  $\bar{\mathbf{h}}_\mathbf{i} = x_{i+1} - F_{k->k+1}(x_i, u_i, p_i)$ ▷ Build discrete equality constraint stage wise
end for

than the MPCs horizon and the planned route length is defined with the variable S in meters.

3.3.4 Solvers used for MPC with rule based speed limitation

The solvers used for in the MPC with rule based speed limitation are SLP and SQP algorithms defined in 1 and 2. The linear program solvers used in the SLP are Matlabs linprog. The quadratic solver used in the SQP is Matlab's quadprog. In addition the nonlinear solver IPOPT is used. The MPC are designed as Matlab functions that have the following inputs, current states $\hat{x}$, previous solution to the NOCP $\hat{\mathbf{z}}$ and current longitudinal position $\hat{s}$. The current state is the feedback from the plant, the previous solutions are used to warm start the optimization and the longitudinal position is used to find the road ahead parameter information. The output is the optimal trajectories for the states and inputs $\mathbf{z}^*$ for the control horizon N, and the new gears produced by the gear optimization $\gamma^*_\mathbf{e}$.

In the sequential programs, the gear optimization is introduced in each sequential iteration, and the algorithms defined in 1 and 2 are augmented to include a gear optimization step after each new sequential update of the optimal solution z_{i+1}. In algorithm 7 a SLP algorithm with gear optimization is defined. This is the same strategy as the one derived in [6]. The same algorithm can be used for the SQP with an extra evaluation of the hessian of the objective function.

Both the SLP and SQP algorithm can be changed into RTI algorithms by using the method defined in 2.1, the convergence criteria is then suppressed and the algorithm stops after a predetermined sequential iteration limit, i_{max}.
For the MPC using IPOPT as its optimization algorithm the NOCP objective function and constraints is evaluated and optimized directly as nonlinear functions. With this algorithm the sequential method for gear optimization defined in 7 is not possible, since IPOPT is not accessible during its internal iterations, and the gear parameter $\gamma_\mathbf{e}$ can not be updated. Instead, gear optimization has to be done after IPOPT has converged, and the IPOPT algorithm has to be called again after one gear optimization.

3.3.5 MPC with lateral dynamics

The lateral dynamics from equation 3.29 and their related constraints 3.30 and 3.31 are too computationally expensive to include in the long horizons needed for energy management. To address this problem the lateral dynamics are simplified with the

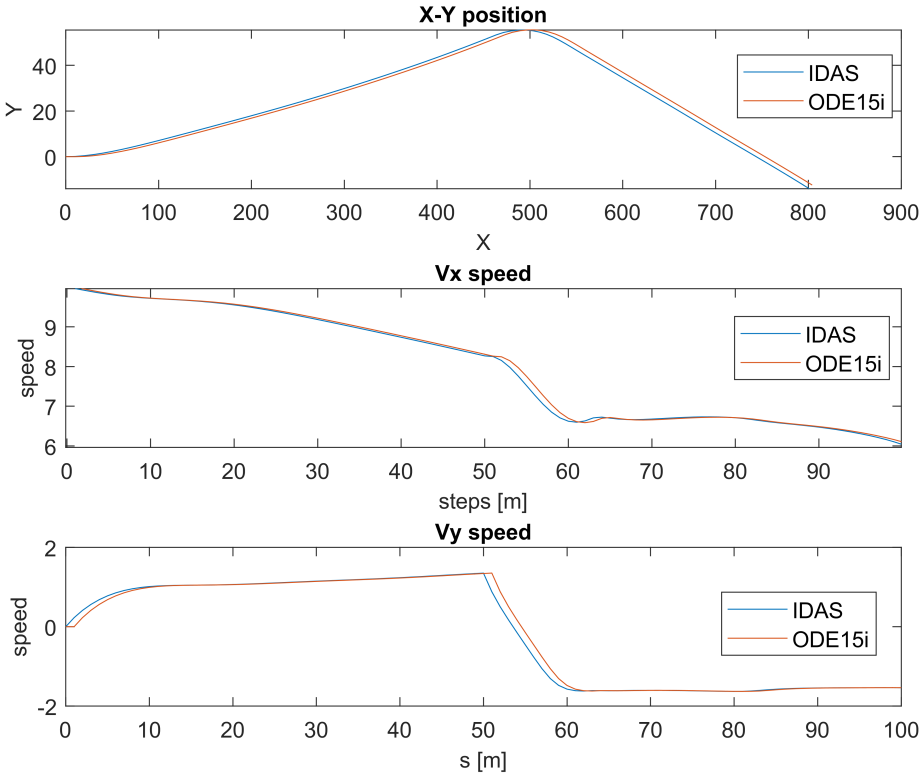

Figure 3.7: Simulations of the lateral dynamics using CasADi IDAS with 1 m step size and Matlabs ode15i with variable step size. For a two turn road with no incline.

methods defined in [7]. The simplification is aimed to handle the equality constraint jacobian $\mathbf{J_{h,lat}}$ w.r.t the lateral dynamics equality constraint $\mathbf{h_{lat}}$, that is built with the method defined in 6. This jacobian is expensive to evaluate since it contains the derivative of $(N\,n)\cdot(N\,n\,m)$ implicit integration steps, where n is the number of states and m is the number of inputs. To avoid evaluating this jacobian in the online controller, the jacobian is constructed offline with the optimal trajectories for the planned route S derived from optimization with a rule based speed constraint. The resulting jacobian is denoted $\bar{\mathbf{J}}_{\mathbf{h,lateral}}$, and can be used inside the sequential programs. Due to the stiffness of the lateral dynamics equations a small stepsize is needed when discretizing, the stepsizes are in the range $0.5 - 2[m]$.

The NOCP including lateral dynamics and stability constraints consist of 12 equality constraints and 21 inequality constraint in each stage, and is built using the methods defined in 3.3.1. Due to its size and complexity only the SLP is used for the MPC including lateral dynamics since it is the fastest method derived in this thesis, as is shown in 4.2.1.

Algorithm 7 Sequential linear program with gear optimization

$\mathbf{z_i} \leftarrow Initial\ guess, i = 1$ ▷ External initial guess
$\gamma_\mathbf{e} \leftarrow \gamma_{\mathbf{e,ref}}$ ▷ Gears from 3.2.4.2
while $RMS(\mathbf{z_{V_x,i+1}} - \mathbf{z_{V_x,i}}) < \epsilon$ **do** ▷ Convergence criteria
  $\bar{\mathbf{f}} = \mathbf{J_f}(\mathbf{z_i}, \mathbf{z_{i+1}}, \gamma_\mathbf{e}, \mathbf{p})$ ▷ Linearized objective function
  $\bar{\mathbf{g}} = \mathbf{g}(\mathbf{z_i}, \gamma_\mathbf{e}, \mathbf{p}) + \mathbf{J_g}(\mathbf{z_i}, \mathbf{z_{i+1}}, \gamma_\mathbf{e}, \mathbf{p})$ ▷ Linearized inequality constraints
  $\bar{\mathbf{h}} = \mathbf{h}(\mathbf{z_i}, \mathbf{p}) + \mathbf{J_h}(\mathbf{z_i}, \mathbf{z_{i+1}}, \mathbf{p})$ ▷ Linearized equality constraints
  $\mathbf{z^+_{i+1}} = LinearProgram(\bar{\mathbf{f}}, \bar{\mathbf{g}}, \bar{\mathbf{h}}, \mathbf{lb}, \mathbf{ub})$ ▷ Solve with suitable solver
  $\mathbf{z_{i+1}} = \mathbf{z_i} + \alpha\,(\mathbf{z^+_{i+1}} - \mathbf{z_i})$ ▷ Update optimal solution with step length α
  $\gamma^+_\mathbf{e} = GearOptimization(\mathbf{z_{i+1}}, \gamma_e)$ ▷ Find optimal gears for trajectory $\mathbf{z_i}$
  $\gamma_\mathbf{e} = \gamma^+_\mathbf{e}$ ▷ Update gears for next iteration
end while
$\mathbf{z}^* = \mathbf{z}$ ▷ Output optimal decision variables
$\gamma^*_\mathbf{e} = \gamma_\mathbf{e}$ ▷ Output optimal ICE gears

3.3.6 Online shrinking horizon (SLP,SQP)

To address situations when the end of the planned route S is closer than the current position s and a control horizon N ahead, a method for online shrinking off the control horizon is defined for the sequential programs. The functions used to build the NOCP are created offline as defined in section 3.3.1 with a fixed horizon, for this section refereed to as N_0. These functions can not be changed when the controller is online, instead the algorithm from 7 is augmented so that only parts relevant for the remaining road of the NOCP function output is passed to the linear or quadratic programs. This method is defeind in algorithm 8, where s is the current position along the planned route, and κ is the number of inequality constraints in one stage, $\kappa = 12$ for the energy management problem, and $\kappa = 21$ when lateral dynamic constraints are included. The variable n is the number of states in the dynamic model, 3 for the simple longitudinal dynamics and 12 for the complex lateral and longitudinal dynamics, m is the number of inputs, 4 for both models.

Algorithm 8 Shrinking horizon algorithm for (SP)

if $s + N_0 \geq S$ **then** ▷ If remaining route is shorten than the original horizon N_0
  $N_s = S - s$ ▷ Length of the remaining route
  $j \in [1, n + (m+n)N_s]$ ▷ Indices off new decision variable $\mathbf{z}$
  $\bar{\mathbf{f}} = \bar{\mathbf{f}}_{i,j}, i \in [1, (n+m)N_s]$ ▷ Objective function for remaining route
  $\bar{\mathbf{g}} = \bar{\mathbf{g}}_{k,j}, k \in [1, \kappa N_s]$ ▷ Inequality constraints for remaining route
  $\bar{\mathbf{h}} = \bar{\mathbf{h}}_{r,j}, r \in [1, nN_s + 1]$ ▷ Equality constraints for remaining route
end if
... ▷ As in algorithm 7

4

Results

4.1 Controller architecture

The architecture of the controller in this paper is illustrated in Figure 4.1. As shown in the figure, the controller architecture consists of three main parts: the inputs form users, the offline controller settings, and the online controller. These three parts will be elaborately detailed in this section.

The inputs from users will be introduced firstly as it is the beginning of the controller architecture. There are three modules with inputs, the vehicle specifications, which include information about the engine, electric motor, battery and other vehicle-specific parameters which are also used as controller offline setting parameters; the route data, which contain details about speed limits, road grades, curvatures and other factors to enable realistic simulations of various routes; as well as the MPC tuning parameters, which involve the horizon, step size and slack variables etc, which are used essentially for tuning and testing.

In the second part of the control architecture, as much preparations as possible are done to speed up the subsequent online control. The controller operates on a discreteize-then-optimized basis, and the measurement data collected at predetermined time intervals also need be discretized. It is essential to introduce an initial guess provided by Volvo software, which estimates the fuel consumption of a conventional diesel vehicle with the reference of gears, velocity and arrival time. In this stage the rule-based stability constraint is also computed.

The third part of the control architecture is the online control, with the data, initial guess, and optimization problem being provided by the second part. The central block is the CasADi-implemented MPC controller, which is a Matlab function that takes measured states and current position as closed-loop input and produces the optimal state trajectory and force requests. GPS is read independently which provides real-time position data that is crucial for real-world implementation. The plant block is an A-double with a electrical dolly. It receives force and gear requests as inputs. Following that, it then outputs the measured states, which include states, velocity, time and battery state of charge.

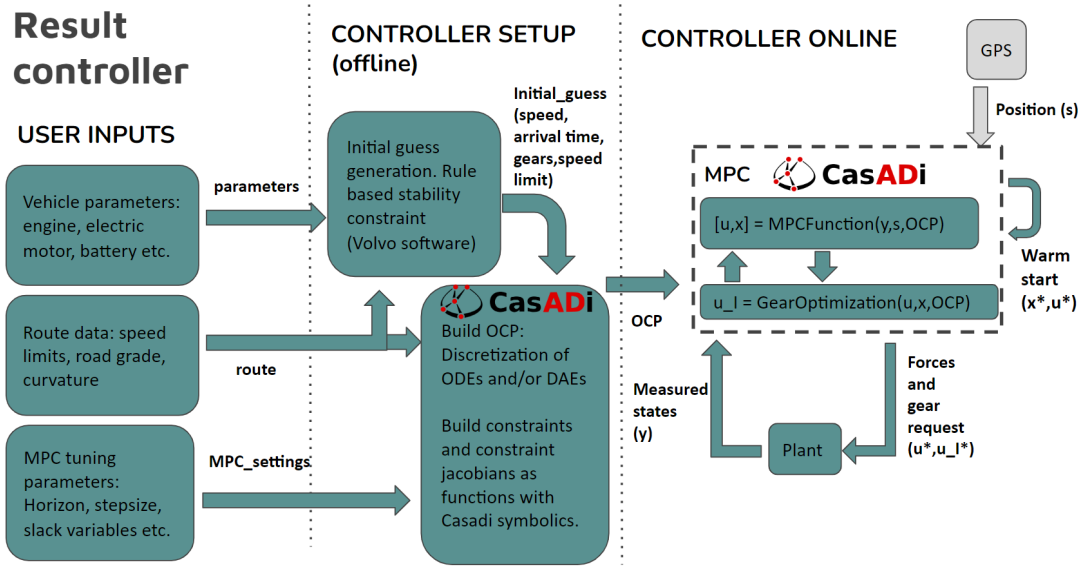

Figure 4.1: Controller architecture.

4.2 Energy management with rule-based speed constraint

In this section an A-double controlled by a energy management MPC with rule-based lateral stability constraint is evaluated by simulations. The fuel consumption of the MPC controlled vehicle is compared with a simulated conventional diesel truck driving the same route. The different optimization algorithms are compared with respect to computation time, robustness and accuracy. In addition, the effect of control horizon N for a moving horizon MPC w.r.t fuel consumption is investigated. Finally, the RTI approach is evaluated.

Important constant vehicle and road parameters are gathered in table 4.1. These parameters are the same for all simulations done in this thesis.

Table 4.1: Table with constant vehicle and road parameters used in all the simulations presented.

Parameter	Value	Symbol
Road friction coeficient	0.8	μ
Combination vehicle total mass [kg]	60250	m
Rolling resistance	0.008	f_r
Wind drag coefficients	9.9840	$\rho_a A_f c_d$
Wheel radius [m]	0.4910	R
Minimum SOC in battery [%]	20	soc_{min}
Maximum SOC in battery [%]	60	soc_{max}

4.2.1 Solution to the NOCP with SLP, SQP and IPOPT

The solution of one NOCP or for a control horizon is the optimal trajectories x^* and u^*. The accuracy and the number of sequential iterations needed for convergence for the SLP and SQP depends on the tunable variables ϵ and α, the SLP is tuned so that the optimal state trajectory x^*_{SLP} is within $RMS(x^*_{SLP} - x^*_{IPOPT}) \leq 0.01$. The SQP is tuned to be as close to IPOPT as possible, but due to convergence issues the result are never as close as the SLP algorithm. The NOCP is discretized with stepsize $ds = 15[m]$ and horizon $N = 333[stages]$, or 5 km.

The optimal speed trajectories for the reference vehicle and the MPCs are shown in figure 4.2. The SLP and IPOPT derive a very similar trajectory while the SQP differs significantly. The simulated road has a legal speed limit of 70 km/h, and for curves in the road the rule-based stability constraints lowers it even further. Due to the low legal speed limit, the conventional vehicle can keep very close to the speed limit for the majority of the route and the resulting arrival time leaves little options for the MPC to change the speed trajectory. Moreover, due to the lower ICE torques needed for the hybrid drive train the gear optimization can find higher gears for many of the stages.

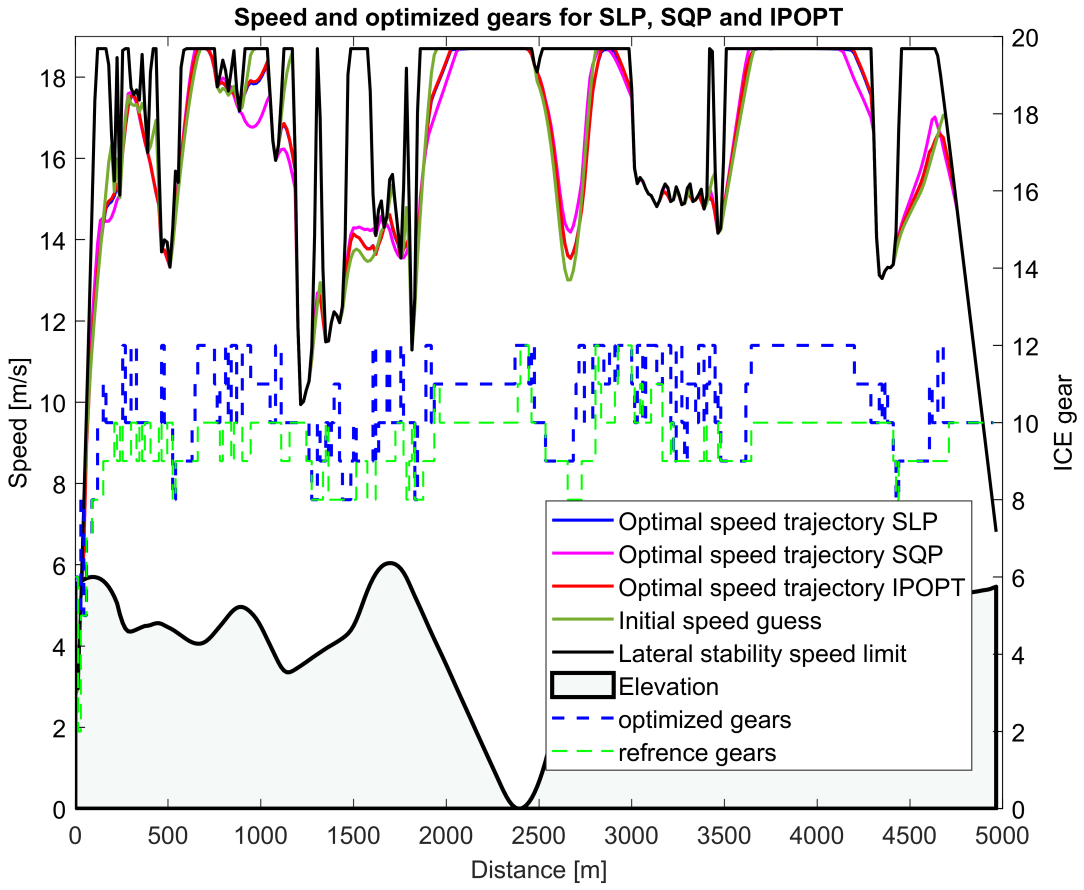

Figure 4.2: Figure illustrating the optimal speed trajectory and ICE gears with rule-based stability constraint for the SLP, SQP and IPOPT for one horizon of 5 km with a stepsize of 15 meters. SLP and IPOPT yield very similar optimal trajectories.

In figure 4.3 and 4.4 the SOC and F_{mw} for the two sequential programs are compared to the trajectories derived by IPOPT. The SLP solution is very similar to IPOPT wile the SQPs differs. All MPCs manages to find trajectories that utilizes the EM to retard and propel the vehicle depending on the road elevation, this is visualised by the F_{mw} force switching sign in down and uphill segments. All controllers manages to find trajectories that start and end with the same SOC which is crucial to make a fair comparison with the conventional vehicle.

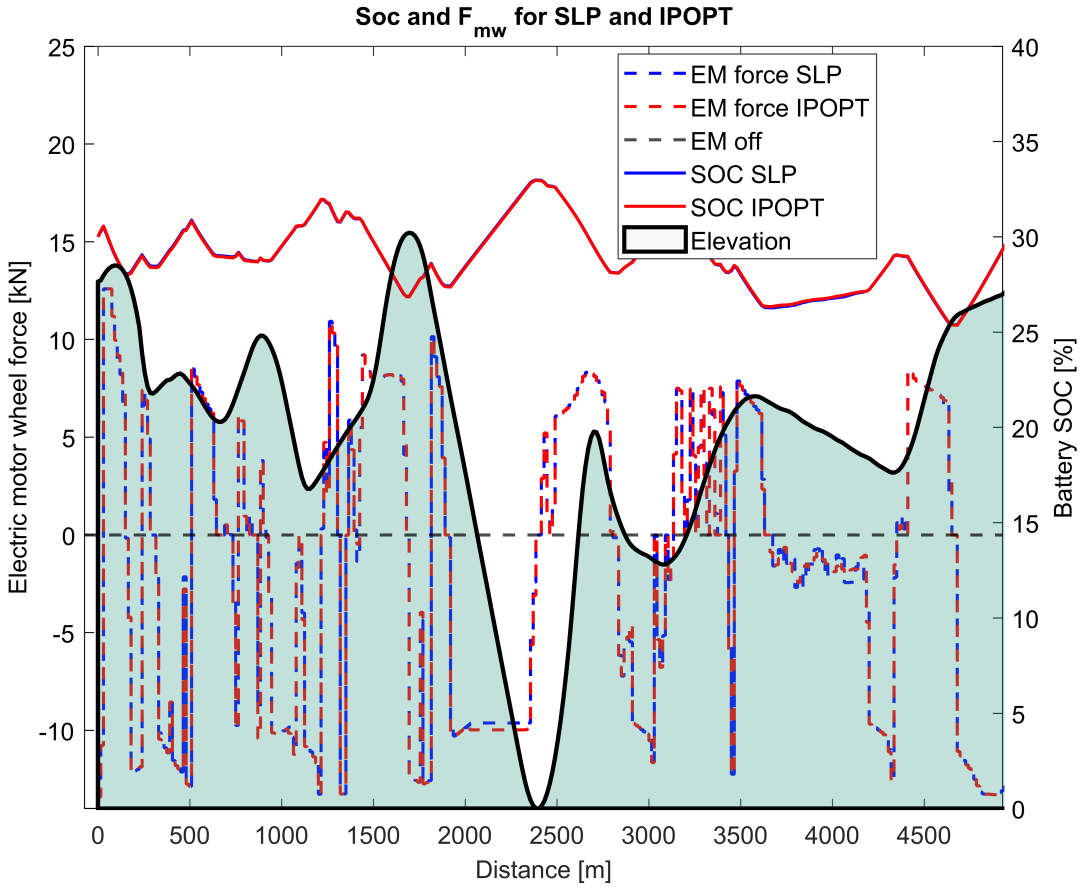

Figure 4.3: SOC and F_{mw} trajectories for the SLP and IPOPT algorithm for one horizon, with route elevation.

In figure 4.5 and 4.6 the speed, F_e and F_{mw} trajectories for the sequential programs and IPOPT are compared. Here again SLP and IPOPT have similar trajectories while the SQP diverges. The forces F_{mw} and F_e work in unison and the optimal trajectories do not involve them acting in opposite directions at the same stage.
Performance data for the different solvers are displayed in table 4.2. Evaluating the results achieved with the SLP shows that its both much faster and more accurate then the SQP. Because of this reason, only the SLP and IPOPT will be used for the further tests conducted in this thesis.
The fuel consumption needed for each of the trajectories is calculated with F_f and compared to the conventional diesel vehicle. The results are shown in table 4.3. In the table the fuel consumption with and without gear optimization is also included. The lower fuel consumption achieved with the SQP is only a result of the few

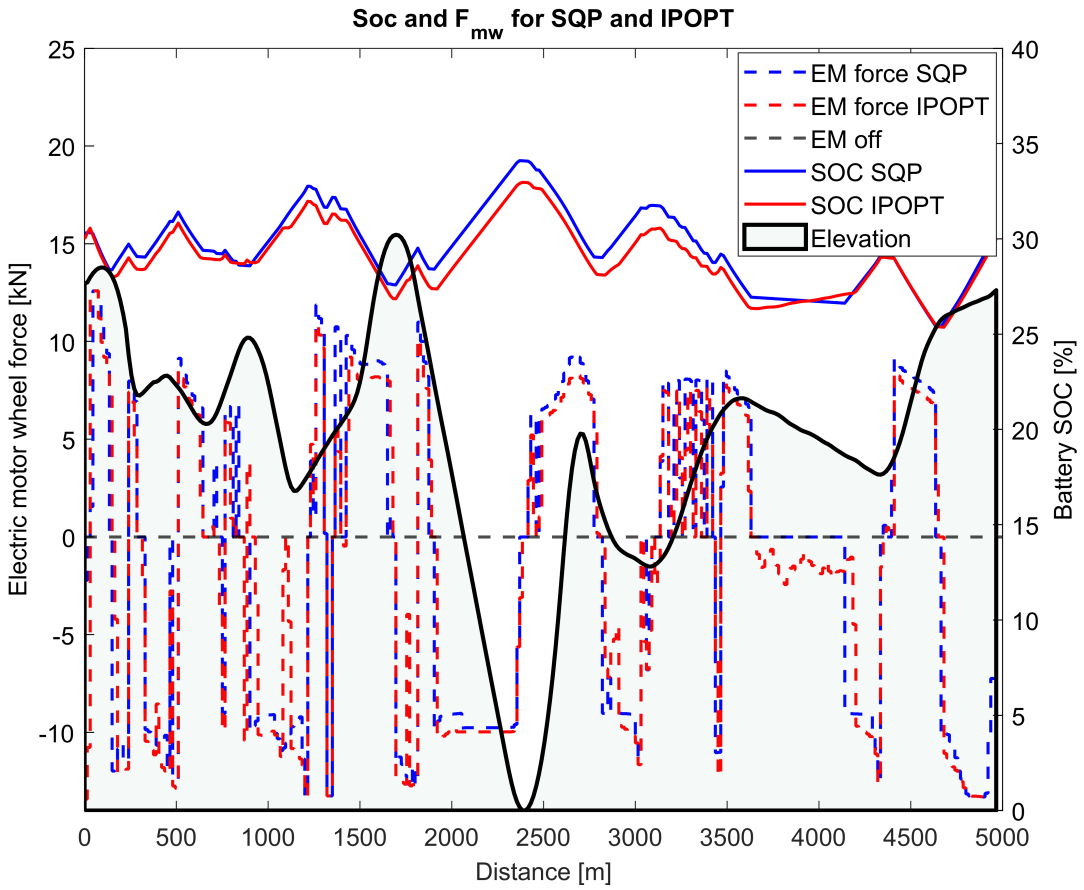

Figure 4.4: SOC and F_{mw} trajectories for the SQP and IPOPT algorithm for one horizon, with route elevation.

sequential iterations done in the algorithm, few iterations means that it is a worse approximation of the nonlinear equations used in IPOPT, and the calculated fuel consumption is not feasible. The gear optimization contributes to decreasing the fuel consumption by 0.15 liters, and the computation time is 0.025 seconds in each sequential iteration, in total 0.1 seconds for the SLP. The fuel consumption for the SLP and IPOPT optimal trajectories are 27% lower then the conventional vehicle.

4.2.2 Relaxing the reference trajectory

Relaxing the reference trajectory with respect to maximum speed and arrival time gives more freedom to the MPC to find a optimal speed trajectory that deviates from the reference vehicles, the results from these tests are shown in figure 4.7 and 4.8. The relaxation is done in two separate cases: either with respect to arrival time, where the MPC has 10% longer time to complete the route, or with respect to maximum speed allowed for the reference vehicle, 10% lower then the legal limit or speed limit derived by the lateral stability constraint. The resulting fuel consumption and arrival time and computation times is displayed in table 4.4. As shown in table 4.4 relaxing the constraints and increasing the freedom of the optimal trajectories can reduce the fuel consumption further. Relaxing the arrival time constraint

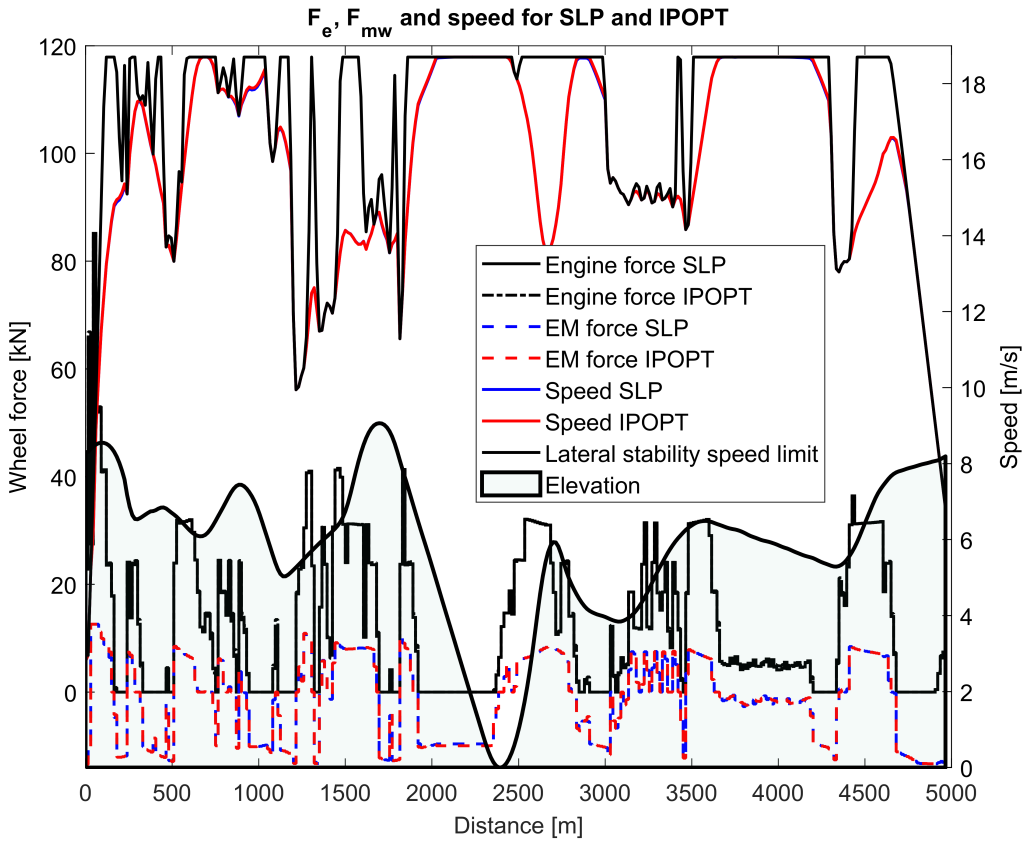

Figure 4.5: Optimal trajectories for SLP and IPOPT for one horizon. Trajectories are speed, F_e and F_{mw}.

lowers the fuel consumption by 35% compared to the fuel consumption for the optimal trajectories with a strict time constraint. And with 53% with respect to the conventional vehicle. Lowering the maximum speed allowed for the reference vehicle also increases its arrival time and lowers its fuel consumption and the optimal trajectories produced for this scenario has a 50% lower fuel consumption. Increasing the MPC freedom makes the problem harder to solve, and the SLP requires more iterations to find the optimal solution.

4.2.3 Moving horizon MPC with SLP and warmstart

To further investigate the MPC using SLP, the closed loop system with moving horizon, and warm starting is simulated and compared to the optimal solution derived by IPOPT. The MPC includes shrinking horizon, and the sequential program is run until convergence in each stage.

The moving horizon MPC speed and SOC trajectories are displayed in figure 4.9 the moving horizon speed trajectory deviates more from the optimal solution derived by IPOPT, but this is expected since the moving horizon system is simulated with an external plant model that introduces numerical error compared to the internal MPC model. The fuel consumption for the simulated moving horizon MPC is 3 liters. The

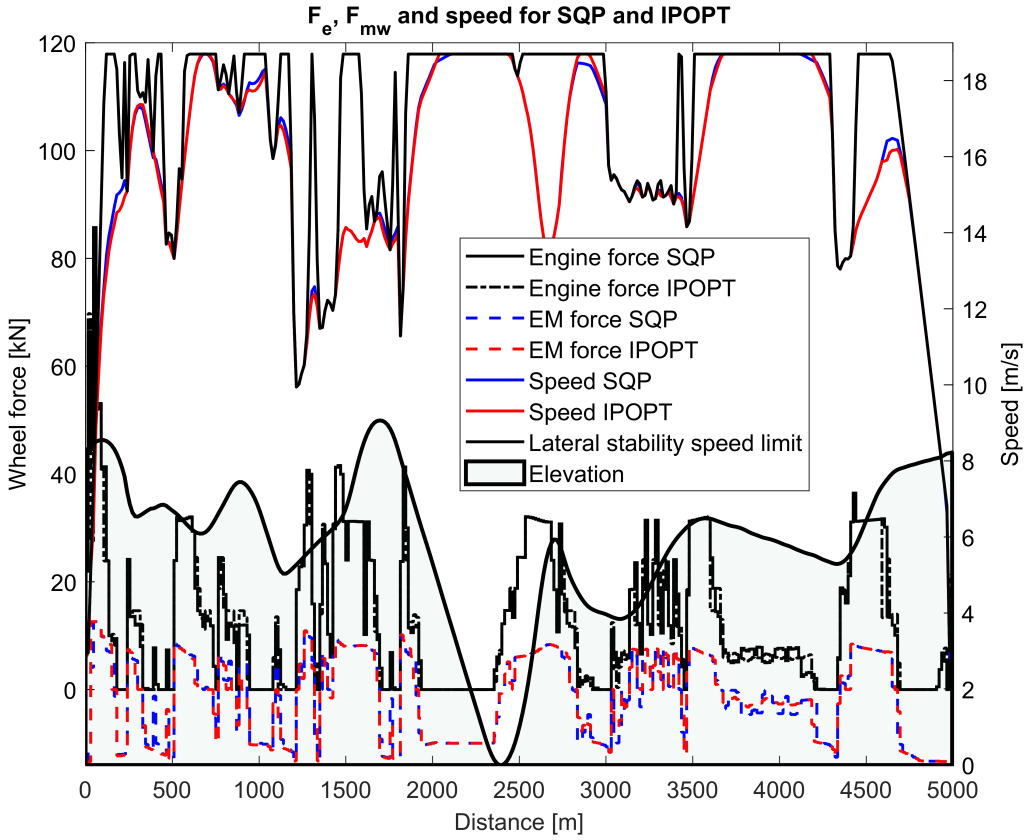

Figure 4.6: Optimal trajectories for SQP and IPOPT for one horizon. Trajectories are speed, F_e and F_{mw}.

MPC has a horizon of 333 stages or 5 km, and sees the end of the planned route from its starting position. The decreased speed limit refrence trajectory was used as a initial guess. Warmstarting the SLP algorithm decreased the computation time by 0.15 seconds.

4.2.4 Effect of control horizon on computation time and fuel consumption

Different control horizons where tested to see how their size relates to fuel consumption and computation time, all controllers include shrinking horizon as they approach the end of the route, 5km away from their starting position. The stage-wise computation times are shown in figure 4.10. Shortening the initial horizon length decreases the computation time, it does however, for the given road, produce a less optimal route trajectory as shown in figure 4.11. This is a expected result since the more information the controller has about its planned route is the better. Shrinking the horizon also has a big impact on SOC management. As mentioned previously the SOC has two constraints, one is the battery lower limit, 20% and one is the end of the road constraint $SOC_0 = SOC_S$ to ensure that energy comparisons are fair. For controllers with short horizons compared to the planned route the battery lower limit can be reached before the end of the route and it might be impossible to return

Table 4.2: Table with computation time data for the different solvers used in the MPC.

	SLP	SQP	IPOPT
Number of sequential itterations	4	2	N/A
Time for LP and QP solution [s]	0.22	12	N/A
Total computation time till convergence [s]	0.85	25.8	30

Table 4.3: Table with fuel consumption used by the optimal trajectories found by the different MPCs and the fuel consumption used by the diesel vehicle, for a 5 km hilly route.

	SLP	SQP	IPOPT	Refrence
Fuel consumption with gear optimization [liters]	4.6	4.1	4.61	6.3
Fuel consumption without gear optimization [liters]	4.71	4.3	4.74	6.3

to the initial SOC in N stages, resulting in the problem becoming infeasible. Because of this there is a trade of between robustness and computation time when designing the controller. The SOC trajectories for the different simulations are shown in 4.12. A solution to this is to constraint every horizon so that the final state of charge is the same as its initial state of charge.

4.2.5 Implementing RTI

In this section the simplified RTI strategy defined in, 3.3 is used for decreasing computation time of the SLP. The results are evaluated and compared to the full convergence SLP and the optimal solution derived by IPOPT. The evaluations are done with respect to solution accuracy and computation time. The horizon length is fixed to $N = 333$ or 5 km. The RTI forced-stop is implemented after $i_{max} = 2, 3, 4$ sequential iterations respectively, denoted RTI_i in the figures. All sequential algorithms have online shrinking horizon as they approach the end of the planned route.

The speed trajectories for MPCs with 3 different RTI settings together with full convergence SLP and IPOPT are shown in figure 4.13. All RTI algorithms manages to stay within the speed limit, but RTI2 diverges from IPOPTs trajectory. The SOC trajectories are shown in figure 4.14, all SOC trajectories diverge from IPOPTs solution but once again RTI2s is the most deviant. The computation time for the RTI_i algorithms and a SLP with full convergence is displayed in figure 4.15. The fuel consumption for the simulated trajectories are displayed in table 4.5. Implementing a RTI algorithm has negative effect on precision compared to full convergence SLP and IPOPTs solution but decreases the computation time significantly.

Table 4.4: Table with fuel consumption, arrival time and computation times for MPCs with relaxed reference.

Relaxed reference speed	SLP	IPOPT	Refrence
Fuel consumption [liters]	2.98	3.0	6.0
Arrival time	359	359	359
Sequential iterations	6	N/A	N/A
Total computation time till convergence [s]	1.24	20	N/A
Relaxed arrival time constraint	SLP	IPOPT	Refrence
Fuel consumption [liters]	2.92	2.95	6.3
Arrival time [s]	362	362	332
Sequential iterations	5	N/A	N/A
Total computation time till convergence [s]	1.0	20	N/A

Table 4.5: Table with simulated fuel consumption for SLP using RTI with forced-stop after 2-4 iterations, compared to full convergence SLP.

	RTI2	RTI3	RTI4	Full convergence SLP
Fuel consumption [liters]	3.15	3.05	3.04	3.0

4.3 Comparison between point mass longitudinal model and multibody lateral and longitudinal model dynamics

The optimal inputs for the rule-based NOCP found by IPOPT, u^*, are used as inputs to the advanced lateral and longitudinal dynamics model. The model is created as a CasADi symbolic function discretized with IDAS with a stepsize $ds = 1m$. The two models are compared on a straight and curvy road, the curvy road is shown in figure 4.16, the same elevation is used for both roads. A speed trajectory comparison between the point mass longitudinal model and the nonlinear multibody lateral and longitudinal model for a straight and curvy roads is shown in figure 4.17 and 4.18.

These simulations show that for a curvy road the simple longitudinal model cant accurately capture the dynamics of the vehicle. This is due to the resistive force acting in the longitudinal direction as a result of steering and nonzero lateral slip. In [7] an approximation of these forces are derived into a single term, according to equation 4.1, where c is a vehicle dependant parameter that needs to be tuned for each vehicle. The term F_c is added to the longitudinal dynamics defined in 3.7 which gives a new longitudinal model equation shown in 4.2.

$$F_c(x, \delta(s)) = c\, v_x\, \delta(s)^2 \tag{4.1}$$

$$h_{v_x,F_c} = \frac{dv_x}{ds} - \frac{1}{mv_x}(-F_{br} + \eta_{te}F_e + F_{mw} + c\, v_x\, \delta(s)^2) \tag{4.2}$$

Including the new extended dynamics in the longitudinal model improves the match between the two models for a curvy road, as shown in figure 4.19. In addition the

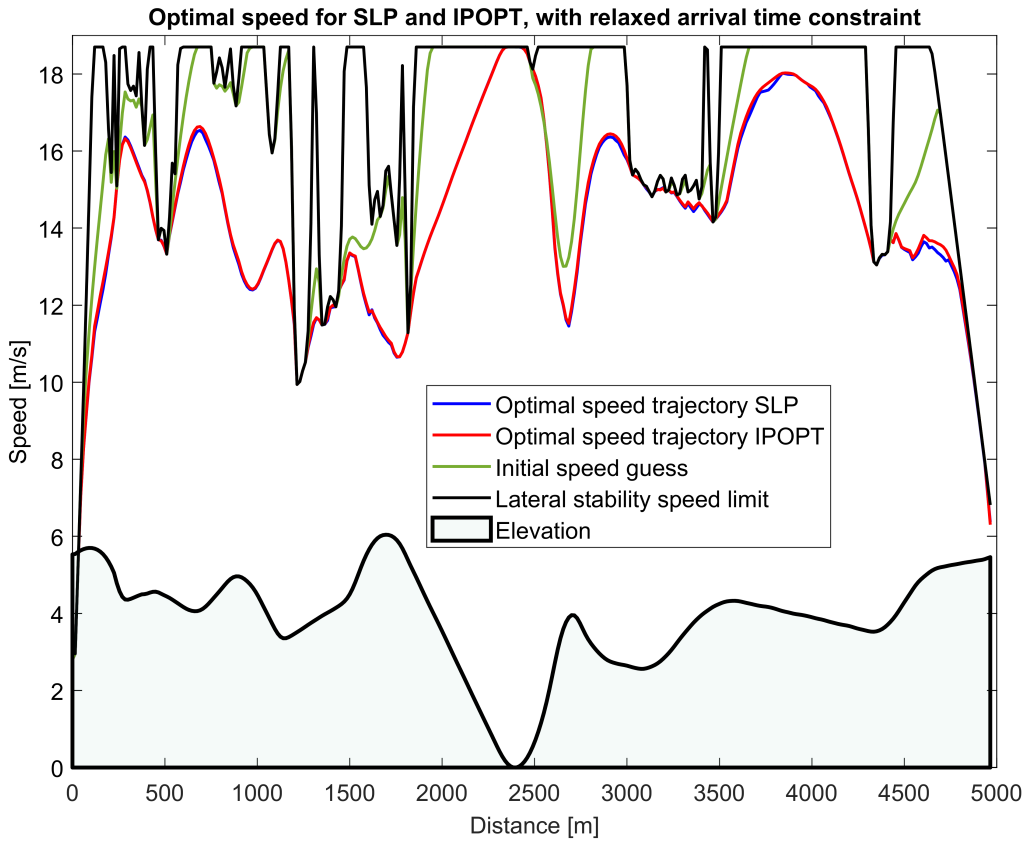

Figure 4.7: Optimal speed trajectory from SLP and IPOPT for one horizon with relaxed arrival time constraint by 10%.

inclusion of additional resistive forces increases the fuel consumption for the optimal trajectories by 1 to 1.5% for the full 5km route.

4.3.1 Evaluating rule-based lateral stability constraint

To evaluate the rule-based lateral stability constraint the additional inequality constraints defined in equations 3.30 and 3.31 are evaluated for the state trajectories simulated with the lateral dynamics, and the optimal inputs derived by IPOPT. The lateral stability constraints are evaluated with a maximum lateral acceleration, $a_{y,max}$ set to $1.5[m/s^2]$ and a safety factor β set to 80%, and the results are shown in figure 4.21 and 4.20. The lateral stability constraint is violated for two instances during the 2.25km route, the wheel force constraint is violated due to sharp braking before a curve.

4.4 Energy management with lateral dynamics constraints.

The NOCP including lateral dynamics is solved with a SLP algorithm using the offline calculated equality constraint jacobian, $\bar{\mathbf{J}}_{\mathbf{h,lateral}}$ defined in 3.3.5. The opti-

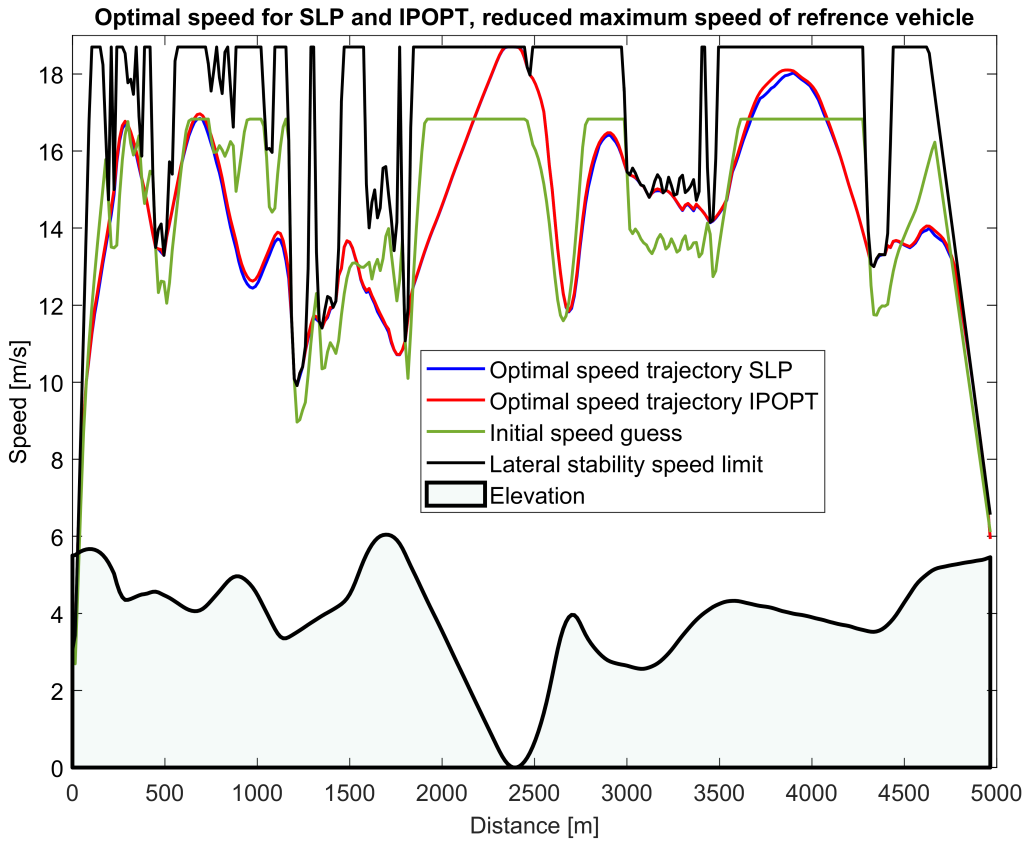

Figure 4.8: Optimal speed trajectory from SLP and IPOPT for one horizon with relaxed refrence speed constraint, 10% lower than lateral stability constraint.

mization is done for a 2km horizon and the discretization stepsize ds is 1 m. The optimal trajectory is compared to the simulated state trajectories and inputs derived with the rule-based speed constraint. The speed trajectories are shown in figure 4.22 with the rule-based speed constraint. The optimal inputs and speed trajectories is displayed in figure 4.23. As is visible from the figure the rule-based lateral speed constraint fails too capture the correct speed limits to guarantee lateral stability. It is either to conservative or too relaxed.
In order to evaluate the model discrepancies created by offline linearization of the equality constraint jacobian the optimal inputs found by the SLP are feed to the IDAS lateral dynamics simulation and the results are shown in figure 4.24. As shown by the figure the trajectories are very similar and the offline linearization does not cause large model discrepancies.

4.5 Improvements to the rule-based speed constraint

In figure 4.21 is is shown that the optimal trajectories produced by the rule-based lateral stability constraint violate the wheel force constraint defined in 3.30. This violation happens when the controller is braking to lower the velocity as its going

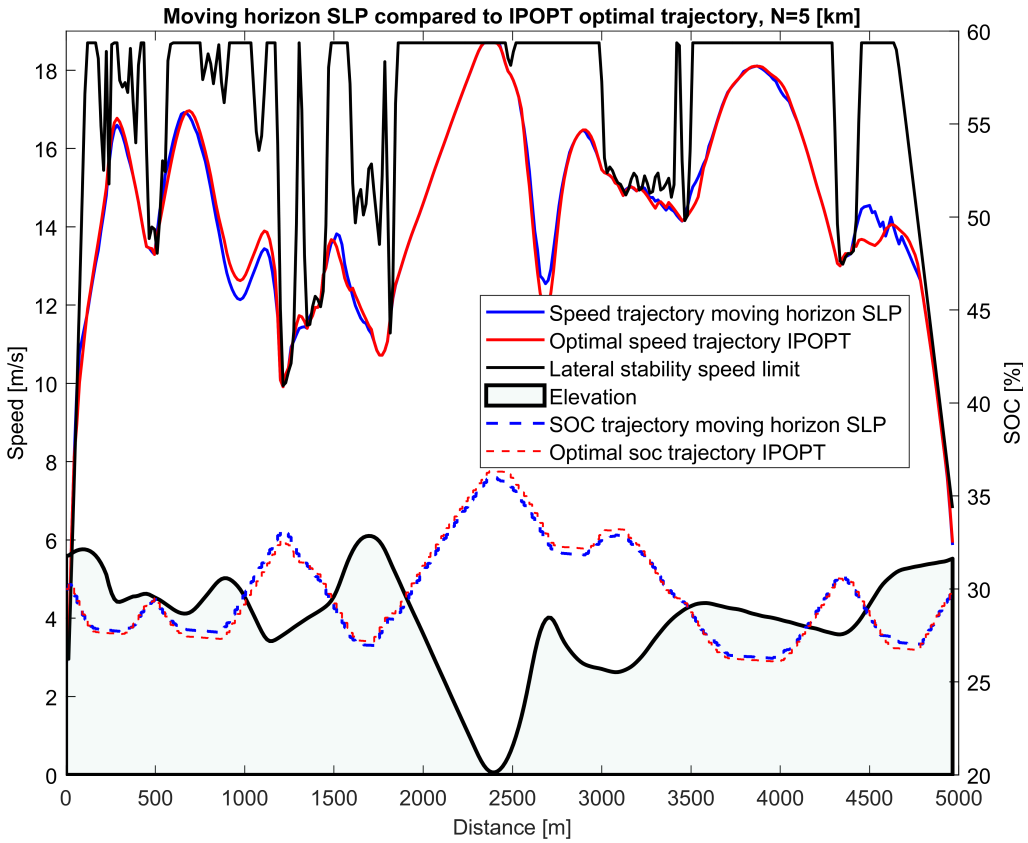

Figure 4.9: Speed and SOC trajectories for moving horizon MPC using SLP, compared to IPOPT optimal trajectories.

into a curve. Since this violation happens on a relatively straight road it is the longitudinal wheel forces F_x that violate the constraint, these forces can be constrained directly for both dynamic models with the mapping matrix A_u defined in 3.27. This results in the following constraint described in 4.3. This constraint in combination with the rule-based speed constraint generates an optimal solution that no longer violates the wheel force constraints on the road defined in 4.16 as shown in figure 4.25. It can not however guarantee that the complete wheel force constraint defined in equation 3.30 is satisfied for all roads, since it does not consider the combination of both longitudinal and lateral wheel forces. Including this constraint does however improve the stability properties further, and this constraint can be tuned to ensure even larger margins with the safety factor β.

$$g_{F_x} = A_u\, u \leq \frac{1}{\beta}\, \mu\, F_z \tag{4.3}$$

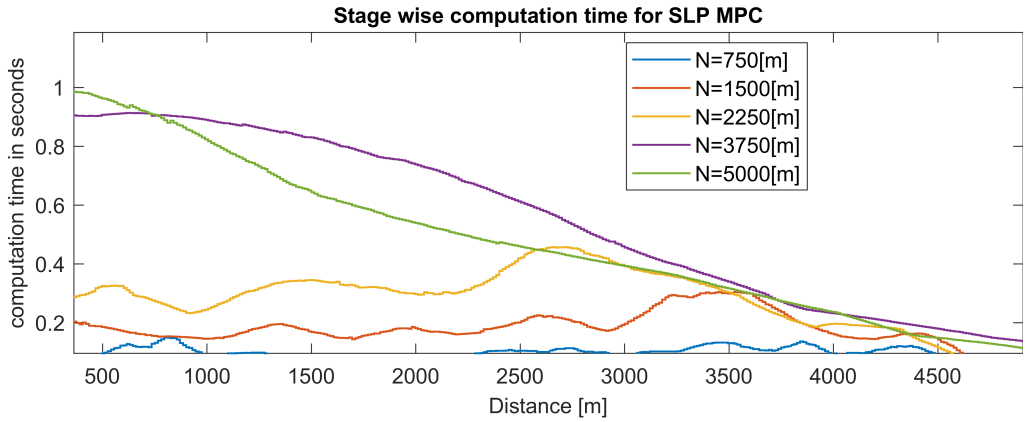

Figure 4.10: stage-wise computation times for SLP MPC with moving horizon for different horizon lengths.

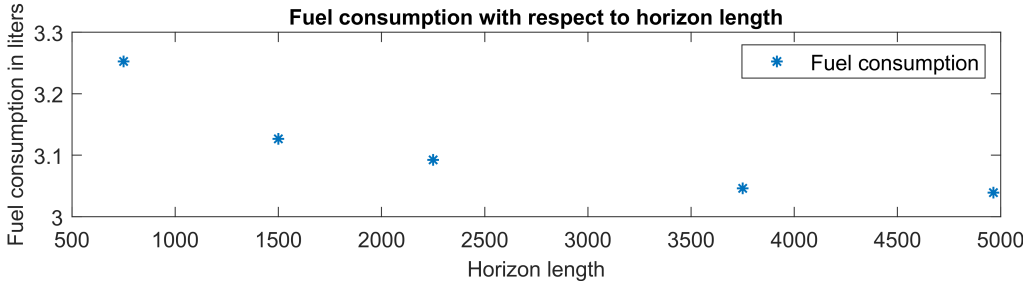

Figure 4.11: Fuel consumption for moving horizon MPC with SLP and different control horizons, simulated for a 5 km route.

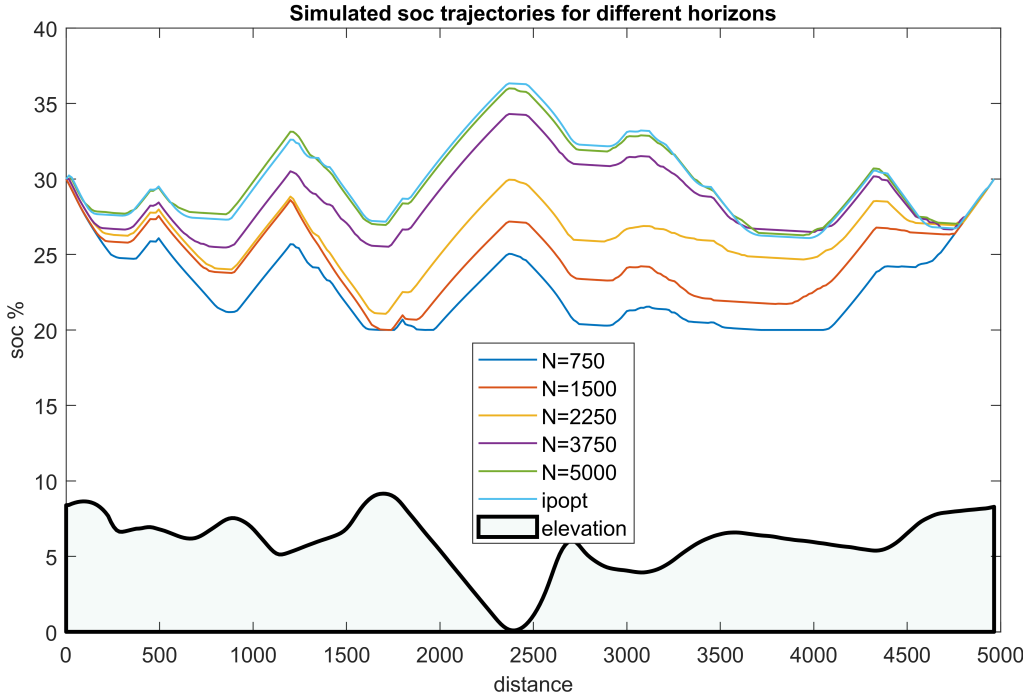

Figure 4.12: SOC trajectories for MPC with SLP and different horizon length.

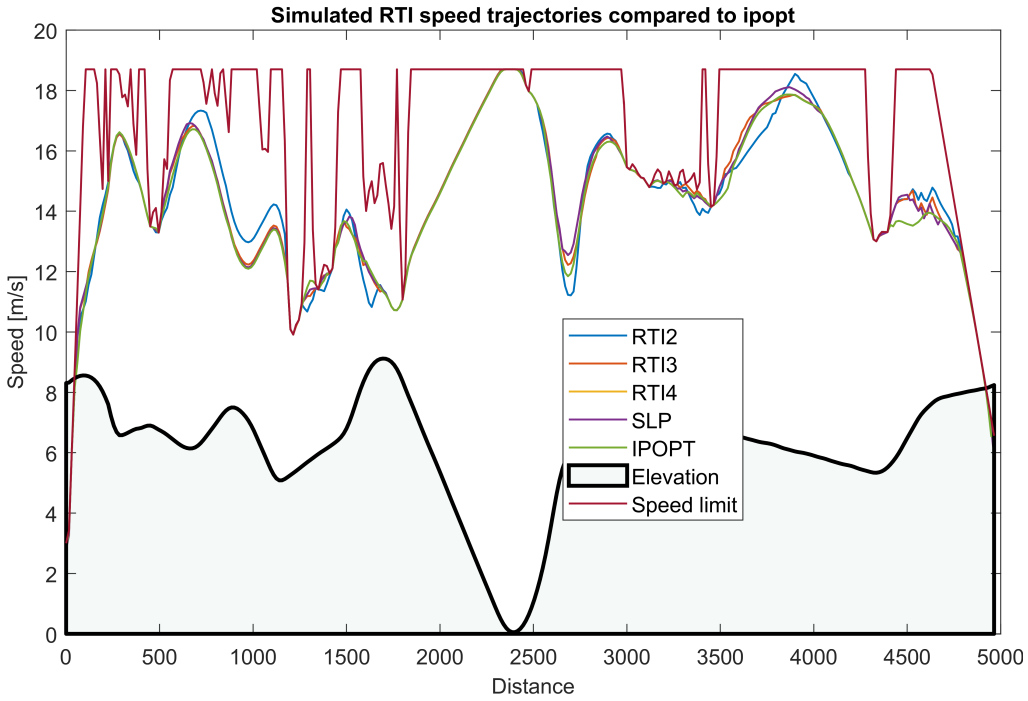

Figure 4.13: Speed trajectory for MPCs including RTI forced stops after 2-4 itteratons ,full convergence SLP and IPOPT. The control horizon is 5 km long. The speed trajectories for RTI4 and the full convergence SLP are very close.

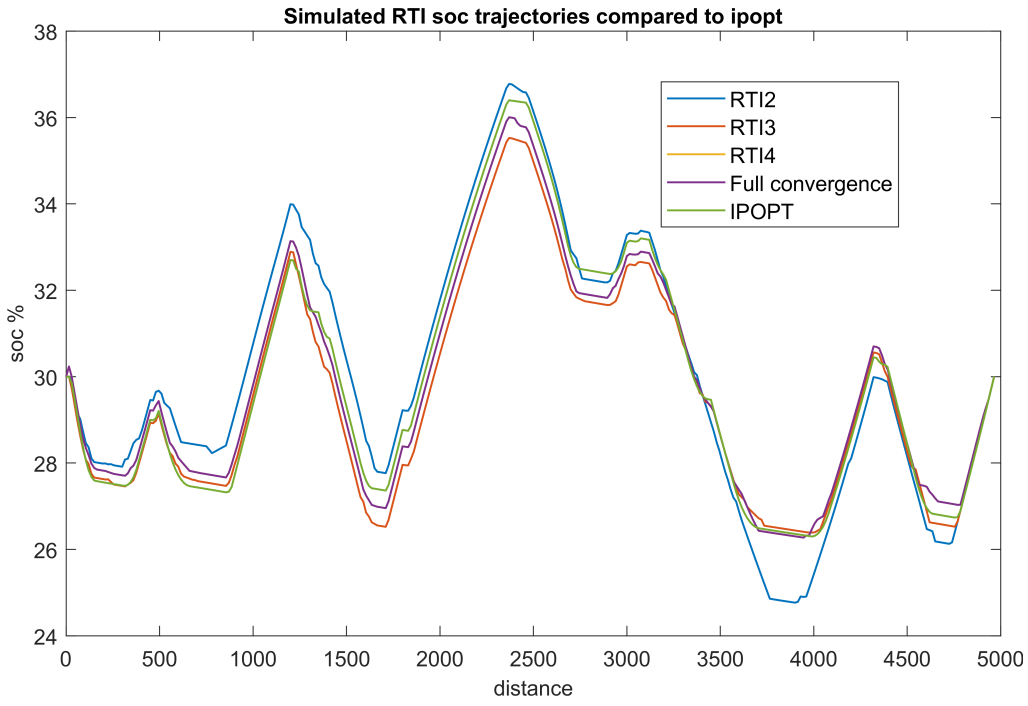

Figure 4.14: SOC trajectory for MPCs including RTI forced stops after 2-4 itteratons, ,full convergence SLP and IPOPT. The control horizon is 5 km long.

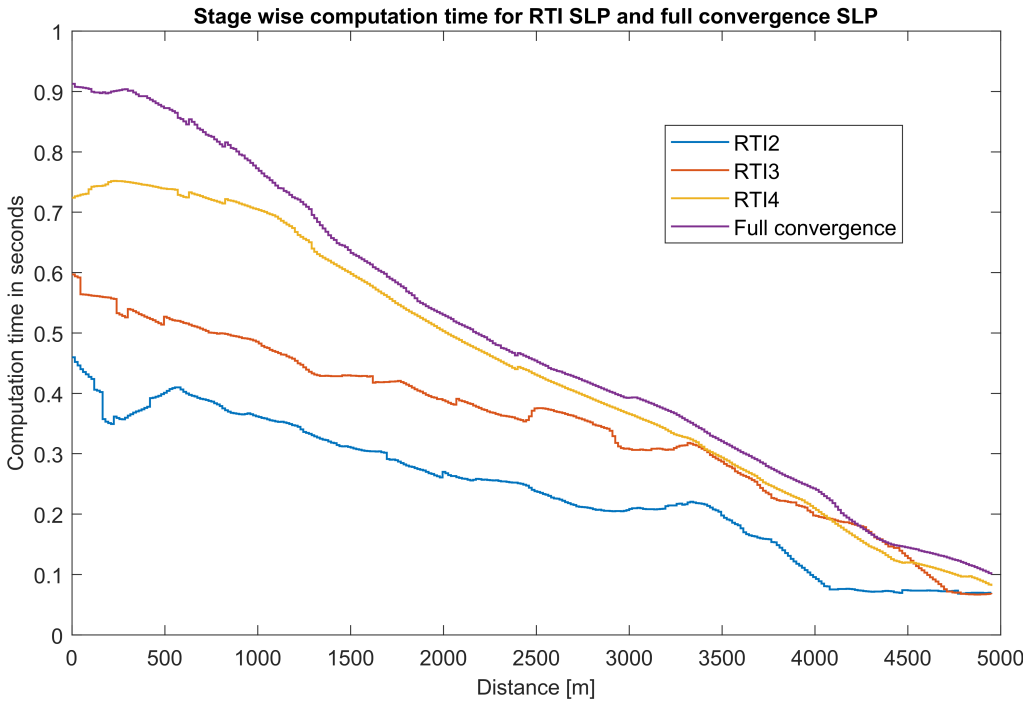

Figure 4.15: Stage-wise computation times for RTI algorithms with forces stop after 2 to 4 iterations, compared to full convergence SLP.

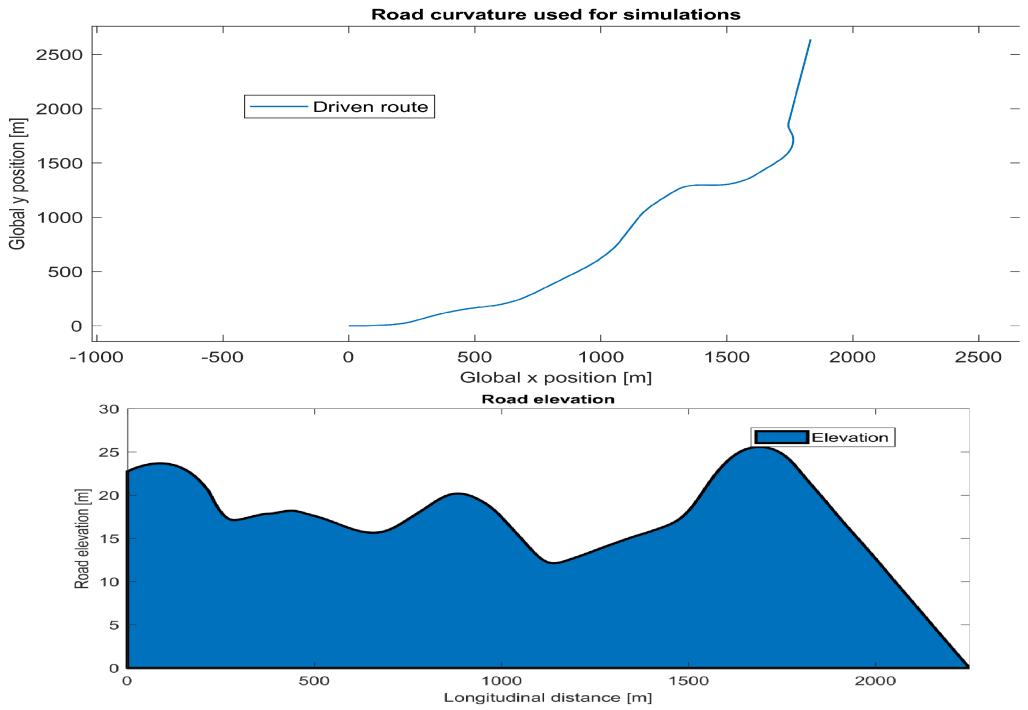

Figure 4.16: Road used to simulate lateral and longitudinal dynamics.

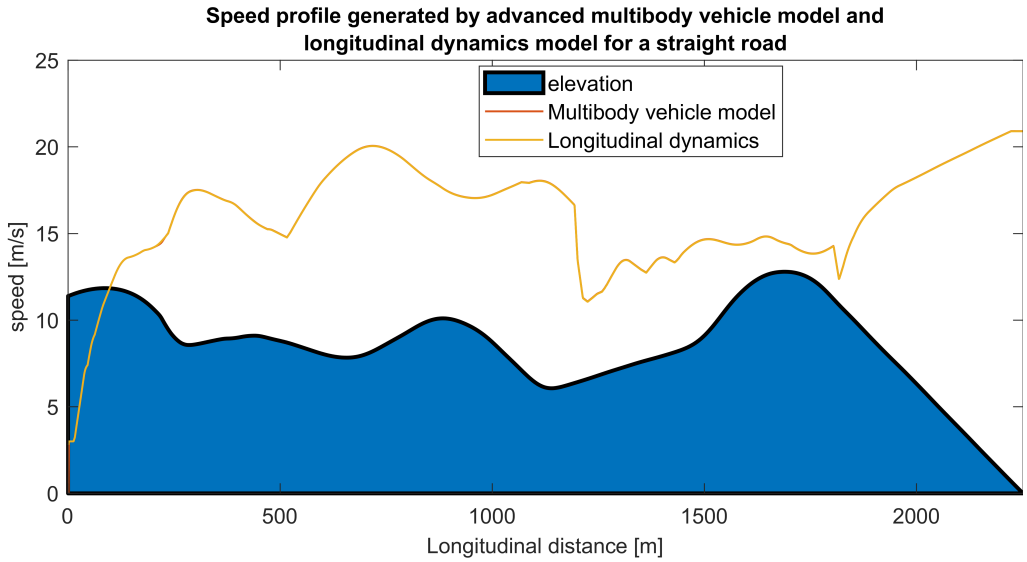

Figure 4.17: Speed trajectory comparison between the point mass longitudinal model and the nonlinear multibody lateral and longitudinal model, for a straight road using the optimal inputs from NOCP using a rule-based speed limit and point mass longitudinal dynamics.

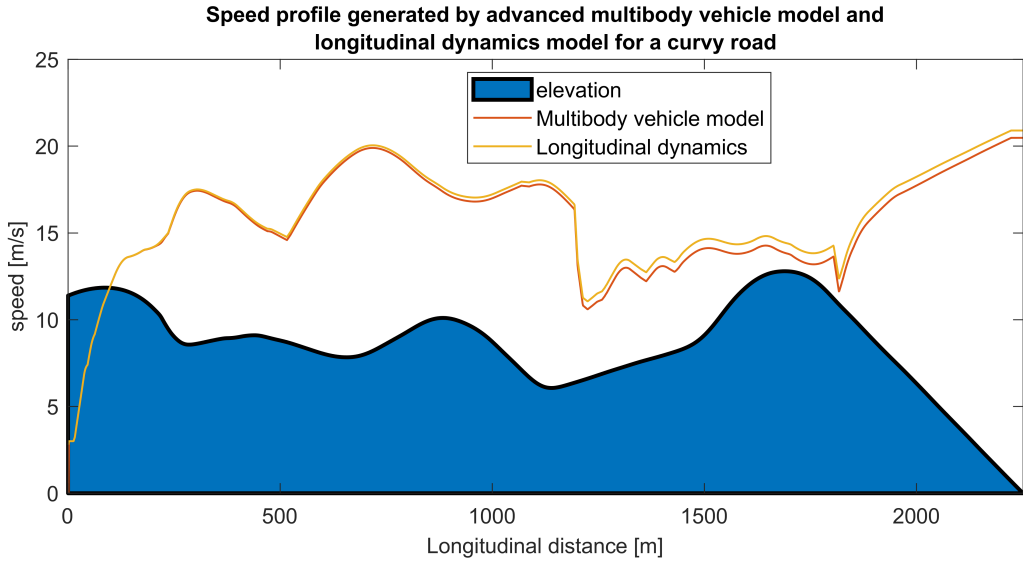

Figure 4.18: Speed trajectory comparison between the point mass longitudinal model and the nonlinear multibody lateral and longitudinal model, for a curvy road using the optimal inputs from NOCP using a rule-based speed limit and point mass longitudinal dynamics.

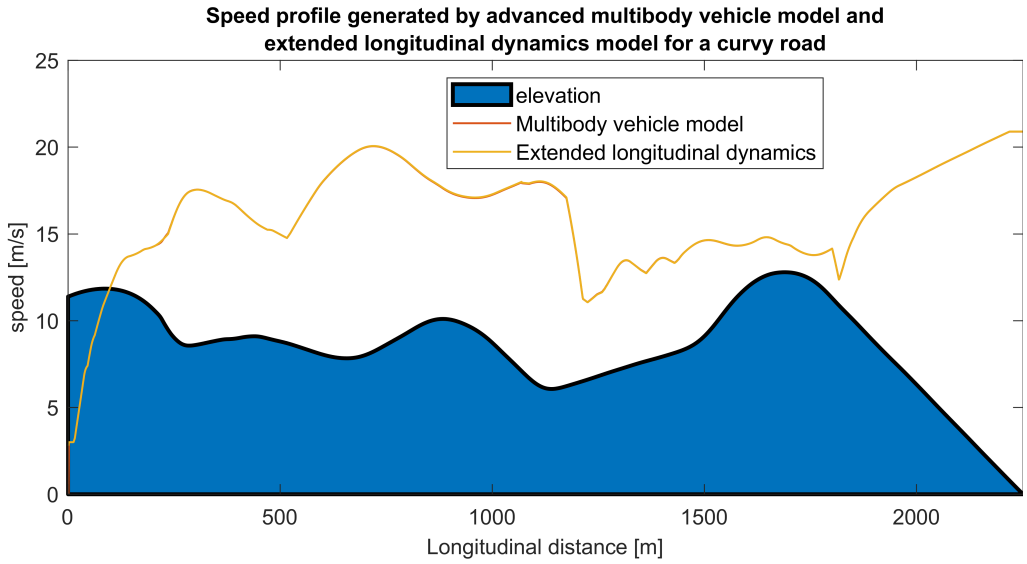

Figure 4.19: Speed trajectory comparison between the multibody dynamic model and the extended longitudinal dynamics model, using the optimal inputs from NOCP using a rule-based speed limit and extended longitudinal dynamics.

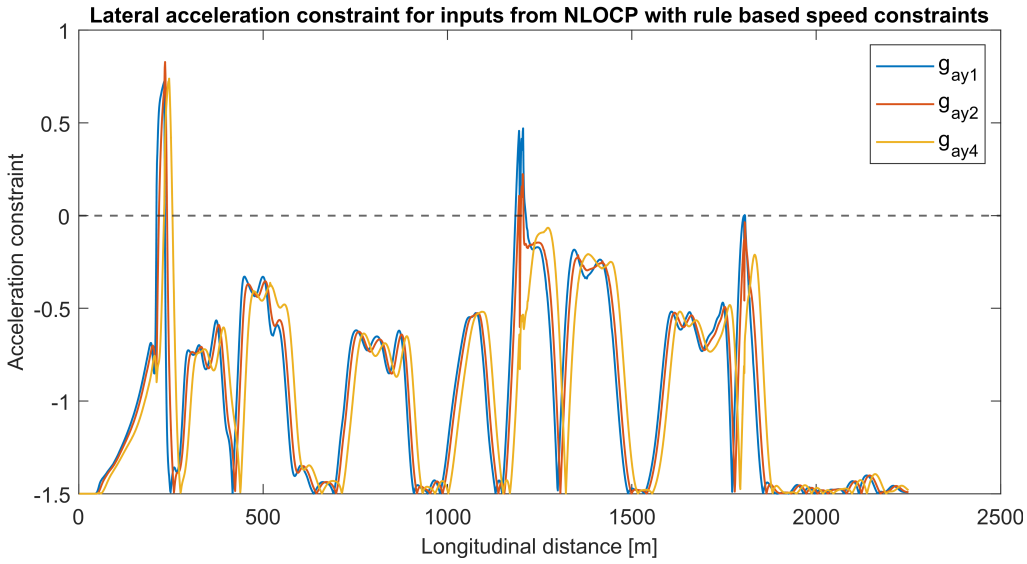

Figure 4.20: Lateral acceleration constraints evaluated for optimal trajectories found for the NOCP using rule-based speed constraint. Index g_{ayi} indicates vehicle unit 1,2 and 4. If the lateral acceleration constraint is larger than 0 the constraint is violated.

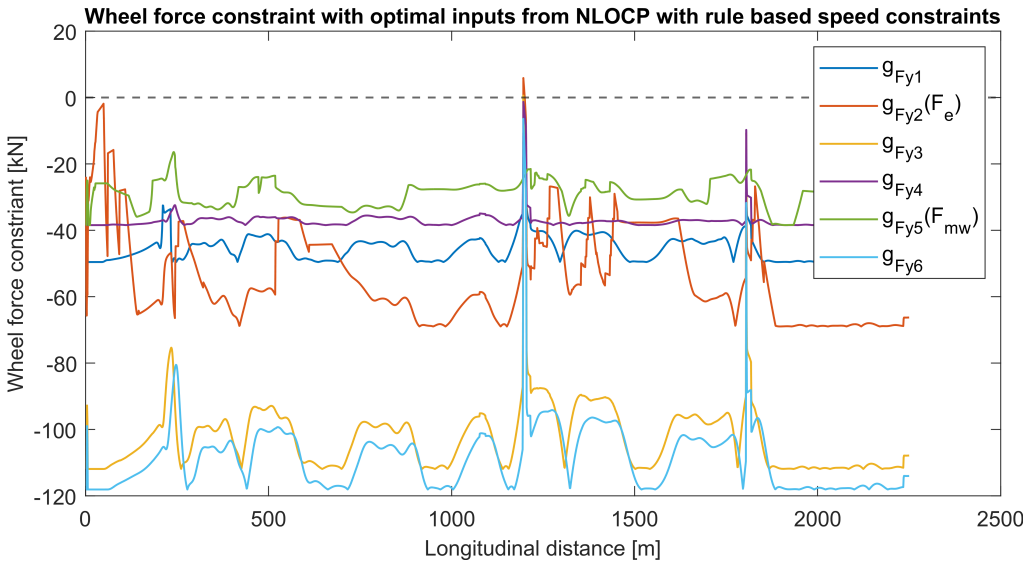

Figure 4.21: Wheel force constraints evaluated for optimal trajectories found for the NOCP using rule-based speed constraint. Index g_{Fyi} indicates wheel group from 1 to 6. With inputs u mapped with A_u matrix defined in 3.27. If the wheel force constraint is larger than 0 the constraint is violated.

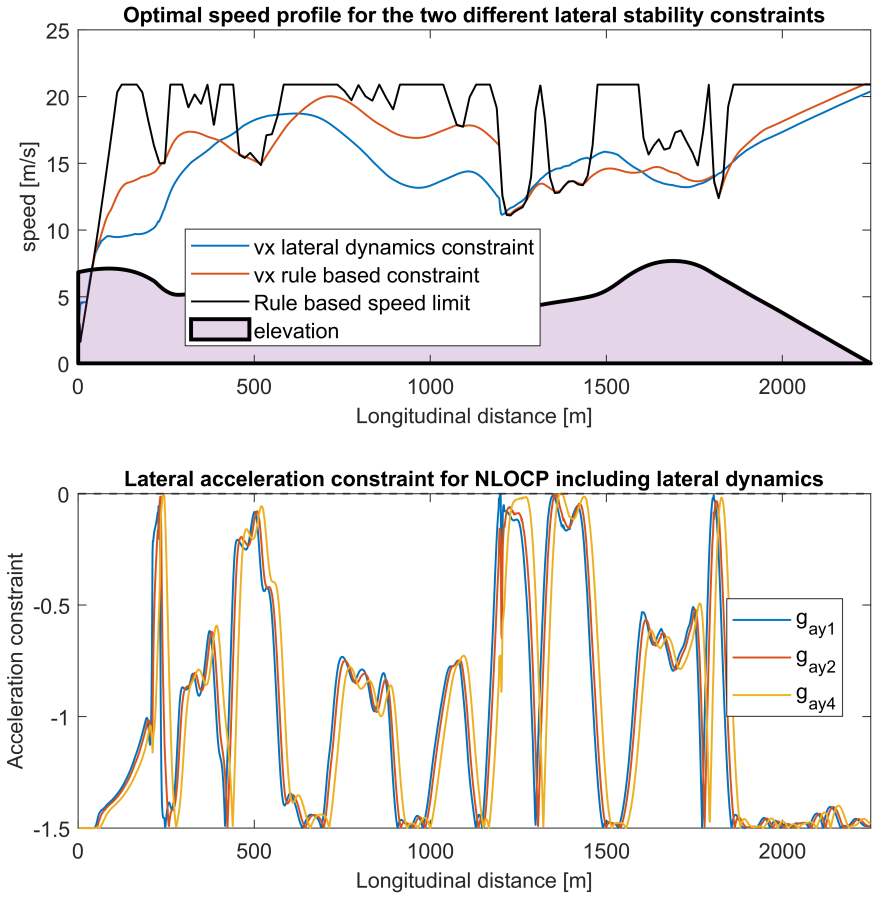

Figure 4.22: The top figure shows the optimal speed trajectory for the NOCP including lateral dynamics compared to the trajectory found with the rule-based speed constraint. The bottom figure shows the lateral acceleration constraint evaluated for the new optimal trajectories.

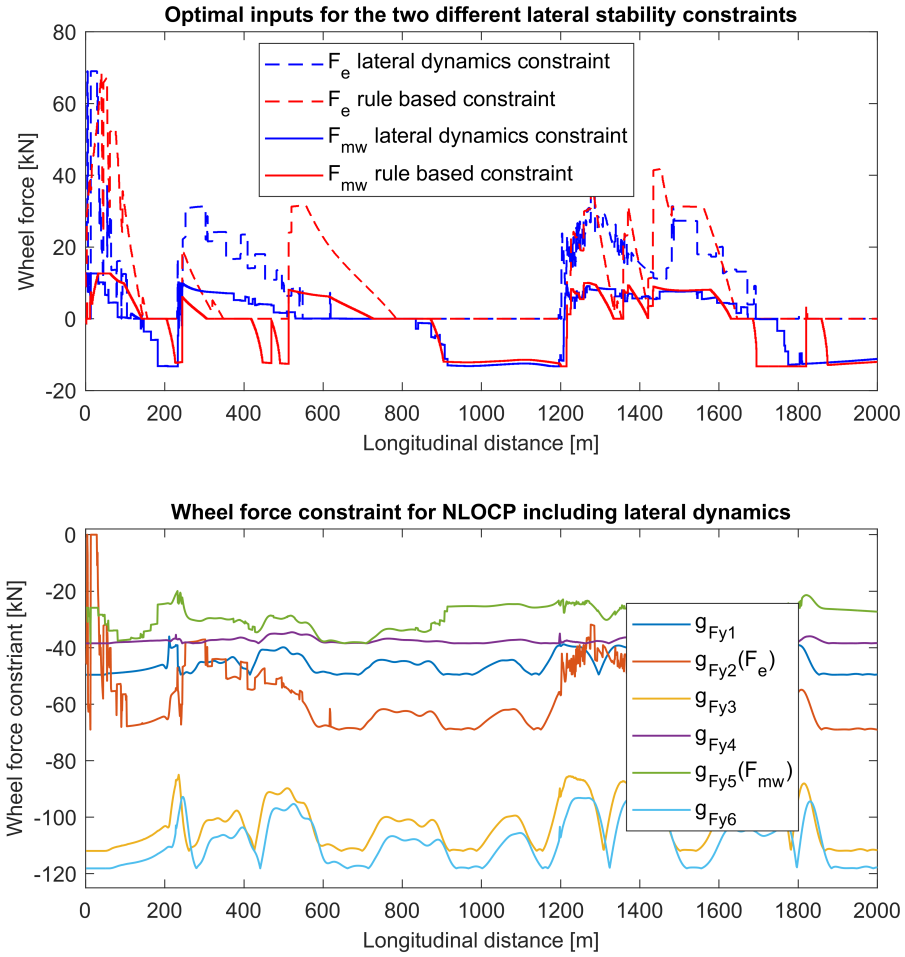

Figure 4.23: The top figure shows the optimal input trajectories for the NOCP including lateral dynamics compared to the trajectories found with the rule-based speed constraint. The bottom figure shows the wheel force constraint evaluated for the new optimal trajectories. Both MPCs utilizes the maximum brake force of the dolly in the curve and the wheel force constraint is not violated.

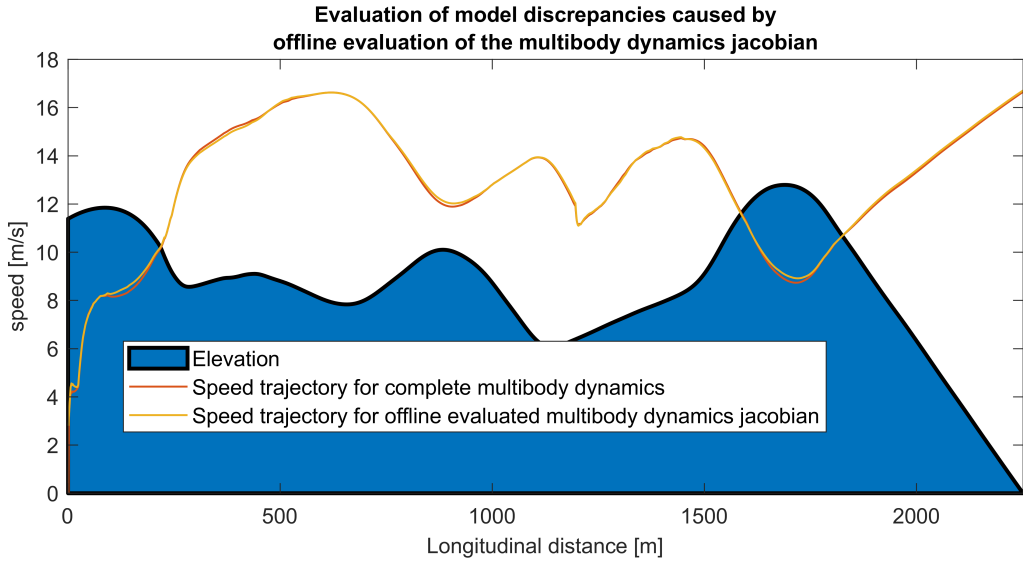

Figure 4.24: Comparison between the optimal speed trajectories derived from the MPC using multibody dynamics with a offline calculated equality constraint jacobian and the complete multibody dynamics simulated using IDAS and the the optimal inputs derived by the MPC.

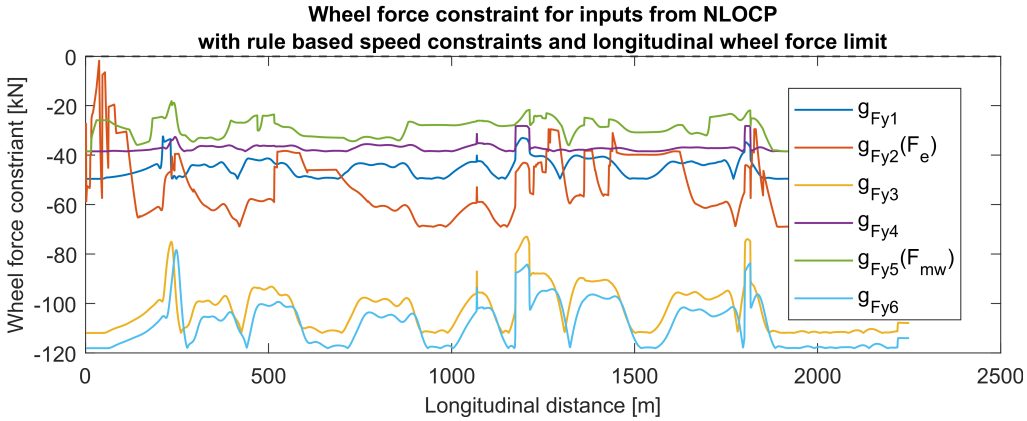

Figure 4.25: Wheel force constraint from equation 3.30 evaluated for the optimal trajectories derived with the rule-based speed constraint and longitudinal wheel force constraint from equation 4.3, for the 6 wheel groups defined in figure 3.3.

5

Conclusion

In this section the results derived in this thesis are discussed and evaluated with respect to the objectives set out in section 1.2.1. Suggestions on further work on the topic is presented.

5.1 MPC design using CasAdi

Using CasAdi to design and implement MPCs generated good results, building the NOCP as symbolic expression and functions makes it very flexible and tuneable. Testing new sparsity patterns, horizon lengths and discretization stepsizes can be done in a single line of code. Building equality constriant functions from continuous dynamics with symbolic discretization using high order Runge kutta methods or implicit integration schemes is a powerful tool that achieves very accurate discrete model dynamics.

The drawback too CasAdi is that even though it is very efficient, the time required to evaluate the large equations is not negligible compared to the solvers computation time. Evaluating the constraint functions and jacobians in the NOCP with rule based speed constraints takes up to 0.1 seconds for a 5 km horizon using a 15m stepsize, a evaluation time that is significant when the total computation time is between 0.8 and 1 seconds. This drawback is even greater for the lateral model dynamics, evaluating the equality constraint for a 2.25 km horizon with a 1 m stepsize takes up to 30 seconds. And evaluating the equality constraint jacobian takes much longer.

It is however possible to further improve the method used to build the NOCP with CasAdi, in [21] CasAdi developers claim that the correct usages, and mix of SX and MX symbolic variables can greatly increase the computation efficiency. The same is true for how the complete horizon functions are built, the CasAdi function.map(N) approach is not necessarily the most computational efficient. It is also possible to speed up the function evaluation by splitting up large symbolic expressions into smaller parts, this is especially relevant for the lateral dynamic equations where the equations are over 50 A4 pages long. CasAdi developers also claim that compiling CasAdi scripts into C code can significantly decrease the computation time.

5.2 Evaluating solvers used in the MPC

When comparing the 3 different solvers used in this thesis the SLP algorithm using Matlabs linprog with a internal simplex solver stands out as the best option out of the 3. A well tuned SLP achieved very similar trajectories to the nonlinear solver IPOPT in only a fraction of the time, it is also sufficiently robust and generates good results for a RTI approach when a 3-4 iteration limit is used, these results are similar to the conclusions drawn in [6]. The SQP algorithm derived in this thesis is not suitable for an MPC application, the optimal trajectories diverge from IPOPTs with similar computation times. The SQP was only tested with Matlabs quadratic solver quadprog and it is possible that other solvers may generate better results, that are closer to the SQP performance in [6]. The nonlinear solver IPOPT is, as expected, too slow to be implemented in real time, it is however possible to improve its computational efficiency. Compiling the solver in to C-code can increase the computation time as well as trying other internal linear solvers.

One important aspect of numerical optimization that was omitted in this thesis is scaling. Scaling the optimization problem reduces the condition number of the involved matrices witch makes the problem easier to evaluate and solve, due to the large difference in numerical values between the decision variables for example soc is in range $[0.2 - 0.8]$ and F_e is in range $[0 - 10^6]$ the condition number of the involved matrices are large. Potential methods for scaling the NOCP for SLP and IPOPT are presented in [9] and [21].

5.3 Lateral stability and dynamic model evaluation

The NOCP with and without lateral dynamics showed that resistive forces caused by the front wheels and nonzero lateral tire slips of all other axles needs to be accounted for to accurately capture the longitudinal dynamics. The inclusion of the F_c term described in 4.1 does not have a large impact on the complexity of the problem, and should be included in the longitudinal model dynamics as suggested in [7].

Simulations comparing the lateral dynamics with and without the offline linearized jacobian also show that this simplification is a valid strategy for simplifying the multibody lateral and longitudinal dynamics. The simplification is however not enough to reduce the computation time so that the NOCP can be solved in real time. Further simplifications is needed in order to include the lateral dynamics.

Comparing the two lateral stability constraints show that the rule based approach is not sufficient for finding a speed limit that guarantees lateral stability. Both the wheel force constraints and the lateral acceleration constraints are violated by the trajectories derived with the rule based constraint, and the optimal speed trajectories that satisfy these constraints are very different. As is displayed by the large

discrepancies between the two speed trajectories in figure 4.22. It is however possible to include a longitudinal force constraint to remove wheel force constraint violation on straight roads, this does however not guarantee lateral stability in curves.

In addition to this it is concluded that the electric motor used in the simulations is not powerfull enough to cause lateral instability. It is however a cause for concern since lateral instability caused by the electric dolly are shown to occur for more powerfull electric motors in [7].

5.4 Evaluation real time implementation

A schematic diagram of how the designed MPC should be implemented is displayed in figure 4.1. The MPC requires an online update from an GPS and data on the planned route. The two critical aspect of implementing the MPC is the computation time T and lateral stability. In order to implement the MPC the computation time should be fast enough to produce a new trajectory at each discretized step, ds. An approximation of the maximum allowed computation time can be derived as $T_{max} \leq \frac{ds}{v_{x,max}} \Omega$ with some factor $\Omega \leq 1$. For the legal top speed of heavy trucks in Sweden 90 km/h and a discretization step size of 15 m the minimum required computation speed would be 0.6 seconds. In the simulations done in this thesis this speed is achievable with either using a RTI3 for a 5 km horizon, or lowering the horizon to 2.25 km for the NOCP with a rule based speed constraint. This is however a rough estimation and the actual computation time needed is heavily dependant on available computational resources on the intended truck.

5.5 Further work

In order to implement the controller in a combination vehicle a new reliable lateral stability constraint needs to be derived. One option is the method derived in [7] where a short horizon including the multibody lateral dynamics is added to the simple longitudinal dynamics equality constraint. The short horizon lateral dynamics will have a smaller discretization stepsize than the simple longitudinal dynamics. This solution is possible to implement in CasAdi and is the natural next step for finding more computationally efficient method for ensuring lateral stability.

Any further work with the goal of finding a more computationally efficient lateral stability constraint can use the trajectories derived by the MPC including both longitudinal and lateral dynamics. The trajectories can be used as a benchmark to verify the effects of further simplifications. Real world tests of the energy management controllers derived in this thesis have been initiated at Volvo Trucks Technology, but due too delays in preparing test equipment the results from these test could not be included in this thesis.

Bibliography

[1] European Commission, (2016), Road transport: Reducing CO_2 emissions from vehicles.

[2] European Commission, (2020), Freight transport statistics - modal split.

[3] Ghandriz, T., Jacobson, B., Laine, L. & Hellgren, J. Impact of automated driving systems on road freight transport and electrified propulsion of heavy vehicles. *Transportation Research Part C: Emerging Technologies*. **115** pp. 102610 (2020)

[4] Ghandriz, T., Jacobson, B., Laine, L. & Hellgren, J. Optimization data on total cost of ownership for conventional and battery electric heavy vehicles driven by humans and by automated driving systems. *Data In Brief*. **30** pp. 105566 (2020)

[5] Research Project in Chalmers. Optimal Distributed Propulsion: Optimal Distributed (over axles) of propulsion for Long Vehicle Combinations (LVCs), (2015-2019).

[6] Ghandriz, T., Jacobson, B., Murgovski, N., Nilsson, P. & Laine, L. Real-Time Predictive Energy Management of Hybrid Electric Heavy Vehicles by Sequential Programming. *Transactions On Vehicular Technology,*. **70** pp. 4113-4128 (2021)

[7] Ghandriz, T. Transportation Mission-Based Optimization of Heavy Combination Road Vehicles and Distributed Propulsion. PhD thesis, Chalmers University Of Technology. (2020)

[8] Ghandriz, T., Jacobson, B., Nilsson, P., Laine, L. & Niklas Fröjd Computationally Efficient Nonlinear One- and Two-Track Models for Multitrailer Road Vehicles. *IEEE Access,*. **8** pp. 203854 - 203875 (2020)

[9] Joseph M. Elble & Nikolaos V. Sahinidis Scaling linear optimization problems prior to application of the simplex method. *Comput Optim Appl,*. **52** pp. 345–371 (2012))

[10] Kharrazi, S., Aurell, J., Sadeghi Kati, M., Jacobson, B., Fröjd, N. & Asp, T. Towards performance based standards in Sweden. *13th International Heavy Vehicle Transport Technology Symposium, San Luis, Argentina*. (2014)

[11] ISO 11992, Road vehicles − Interchange of digital information on electrical connections between towing and towed vehicles. (2003)

[12] Jia, Y., Saito, T., Itoh, Y., Nukezhanov, Y. & Görges, D. Energy-Optimal Adaptive Cruise Control in Time Domain based on Model Predictive Control. *IFAC-PapersOnLine*. **51**, 846-853 (2018), 5th IFAC Conference on Engine and Powertrain Control, Simulation and Modeling E-COSM 2018

[13] Lee, Y., Lee, D., Lee, S. & Kim, Y. A Comparative Study on Model Predictive Control Design for Highway Car-Following Scenarios: Space-Domain and Time-Domain Model. *IEEE Access*. **PP** pp. 1-1 (2021,11)

[14] Zhou, Q., Zhao, D., Shuai, B., Li, Y., Williams, H. & Xu, H. Knowledge implementation and transfer with an adaptive learning network for real-time power management of the plug-in hybrid vehicle. *IEEE Transactions On Neural Networks And Learning Systems*. **32**, 5298-5308 (2021)

[15] Chatrattanawet, N., Kheawhom, S. & Arpornwichanop, A. Robust Model Predictive Control Strategy for LTV and LPV Systems of the Internal Reforming Solid Oxide Fuel Cell. *Computer Aided Chemical Engineering*. **37** pp. 1733-1738 (2015)

[16] Gros, S., Zanon, M., Quirynen, R., Bemporad, A. & Diehl, M. From linear to nonlinear MPC: bridging the gap via the real-time iteration. *International Journal Of Control*. **93**, 62-80 (2020)

[17] Lofberg, J. YALMIP : a toolbox for modeling and optimization in MATLAB. *2004 IEEE International Conference On Robotics And Automation (IEEE Cat. No.04CH37508)*. pp. 284-289 (2004)

[18] Andersson, J., Gillis, J., Horn, G., Rawlings, J. & Diehl, M. CasADi: A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*. **11**, 1-36 (2019)

[19] A. Zanelli, A. Domahidi, J. J. Jerez & M. Morari FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*. **93**, 13-29 (2020)

[20] Andersson, J.,Kozma, A., Gillis, J., Diehl, M.: User Documentation for CasADi . (2018) http://guide.casadi.org.

[21] Andersson, J.,Kozma, A., Gillis, J., Diehl, M.: CasADi developers blog. (2022) https://web.casadi.org/blog/.

[22] Wächter, A. Short tutorial: Getting started with ipopt in 90 minutes. *Dagstuhl Seminar Proceedings*. (2009)

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

[24] General Assembly Envision 2030: 17 goals to transform the world for persons with disabilities. (2015)

[25] Kharrazi, S. Performance based standards for high capacity transports in Sweden. *No.: VTI Rapport 948A*. (2017)

[26] James B. Rawlings, David Q. Mayne, Moritz M. Diehl. Model Predictive Control: Theory, Computation, and Design, 2nd Edition.

[27] Commons, W. File:Jackknifing and Trailer swing.jpg — Wikimedia Commons. (2020),

[28] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, & M. Diehl From linear to nonlinear MPC: bridging the gap via the real-time iteration. *International Journal of Control*. **93** pp. 66 - 80 (2020)

Department Of Mechanics and Maritime Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY