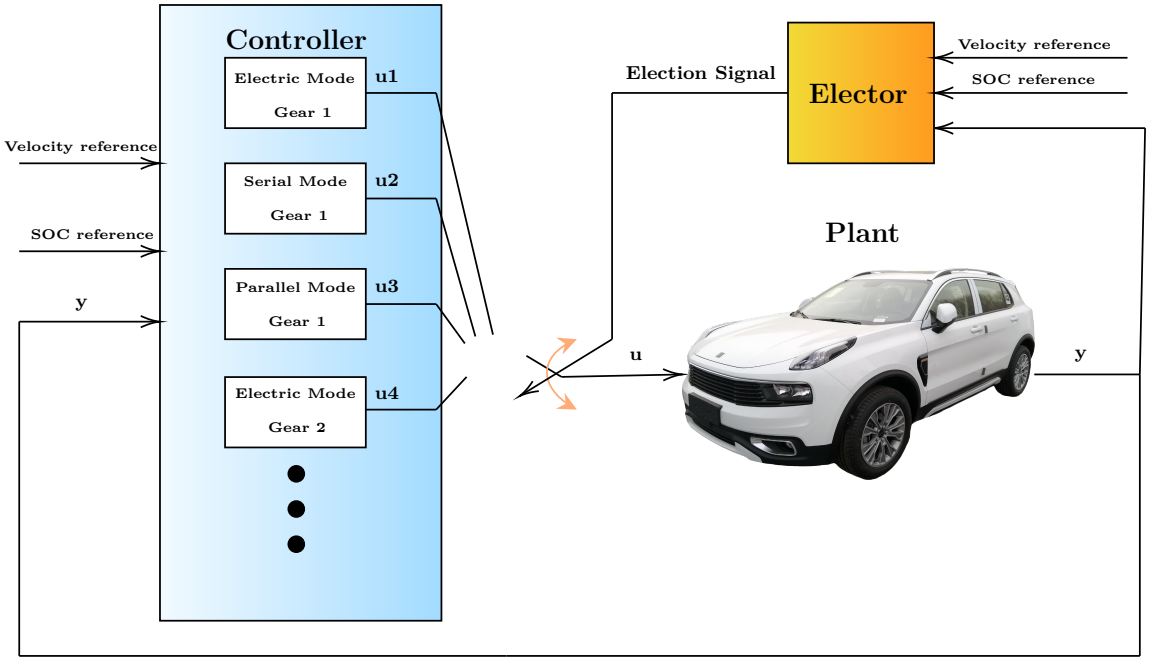
CHALMERS
UNIVERSITY OF TECHNOLOGY



Three-Mode Hybrid Powertrain Optimal Control to Track Offline Optimized References

Nonlinear Model Predictive Control of Switched Systems used to Track References Established with Dynamic Programming

Master's thesis in Systems, Control and Mechatronics

DANIEL HULTGREN, TEODOR HUSMARK

DEPARTMENT OF ELECTRICAL ENGINEERING
DIVISION OF SYSTEMS AND CONTROL

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020
www.chalmers.se

Master's thesis

Three-Mode Hybrid Powertrain Optimal Control to Track Offline Optimized References

Nonlinear Model Predictive Control of Switched Systems used to Track References Established with Dynamic Programming

DANIEL HULTGREN, TEODOR HUSMARK

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2020

Three-Mode Hybrid Powertrain Optimal Control to Track Offline Optimized References
Nonlinear Model Predictive Control of Switched Systems used to Track References Established with Dynamic Programming
DANIEL HULTGREN, TEODOR HUSMARK

Supervisor: Karthik Prasad, CEVT AB
Examiner: Jonas Fredriksson, Department of Electrical Engineering

Master's Thesis 2020
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Illustration of the switched system control concept used to minize energy consumption.

The car photo in the cover image is "Lynk & Co 01 photographed in Beijing, China." by Navigator84, used under CC BY-SA 4.0 / Cropped from original

Typeset in LaTeX using a template created by David Frisk
Gothenburg, Sweden 2020

Three-Mode Hybrid Powertrain Optimal Control to Track Offline Optimized References
Nonlinear Model Predictive Control of Switched Systems used to Track References Established with Dynamic Programming
DANIEL HULTGREN, TEODOR HUSMARK
Department of Electrical Engineering
Chalmers University of Technology

Abstract

This thesis implements a new offline-online coupled powertrain control solution for a three-mode hybrid electric vehicle with the purpose of minimizing the total energy consumption over a route. The offline component uses a simplified version of the powertrain to generate optimized SoC and velocity references through dynamic programming given speed limits and topographic profile of a road segment. To act on the references, the online component employs the optimal control technique of nonlinear model predictive control on a detailed dynamical model of the powertrain by generating torque setpoints to the power sources and selecting the most optimal gear. The coupled solution is also given the feature of treating driver interaction by acting on driver requested torque in overtake scenarios.

The full solution was implemented and tested through simulations in MATLAB and Simulink. The coupled solution showed benefits over a pure online controller due to its strong predictive performance. Compared to solely using the online controller, the coupled solution produces less engine events which is beneficial for drivability, can ensure that regenerative braking segments are utilized and can cope with emission free zones while meeting the control objective.

Further simulation results shows that the developed solution is able to act on driver requested torque in a satisfactory manner and that the tuning parameters of the solution plays a big role in the overall behavior of the coupled solution.

Keywords: Nonlinear Model Predictive Control, Dynamic Programming, Optimal Control, Optimization, Powertrain, Hybrid Electric Vehicle, Switched Systems.

Strategi för optimal reglering av en hybrid Drivlina för att nå energikrav
Olinjär modellprediktiv reglering med växlade system för att följa offline optimerade referenser skapade genom dynamisk programmering på en tre-läges hybrid drivlina
DANIEL HULTGREN, TEODOR HUSMARK
Institutionen för Elektroteknik
Chalmers tekniska högskola

Sammanfattning

Detta projekt implementerar en ny lösning för styrningen av en hybrid drivlina med tre körlägen. Den nya lösningen sammanfogar offlineoptimering med onlinereglering med syftet att minimera den totala energiförbrukningen för en given rutt. Denna optimering använder en förenklad matematisk modell av drivlinan för att genom dynamisk programmering kunna generera optimerade laddningstillstånds- och hastighetsreferenser givet hastighetsbegränsningar och topografisk profil av den aktuella rutten. Onlineregleringen använder sedan olinjär modellprediktiv reglering i kombination med en detaljerad matematisk modell av drivlinan för att agera på referenserna genom att generera börvärden till drivlinans effektkällor samt välja den mest optimala växeln. Den sammanfogade lösningen gavs också möjligheten att realisera förfrågan om vridmoment från föraren vilket kan vara nödvändigt i situationer så som en omkörning.

Helhetslösningen implementerades och testades genom upprepade simuleringar i MATLAB och Simulink. Den sammankopplade lösningen visar fördelar över en enbart onlinereglering tack vare dess starka prediktiva förmåga. Till skillnad från att enbart använda onlinereglering så producerade den sammankopplade lösningen färre lägesbyten vilket är fördelaktigt för körbarheten. Dessutom kan den kopplade lösningen säkertställa att tillfällen för generativ inbromsning alltid utnyttjas samt att utsläppsfria zoner följs utan att bryta mot kontrollmålet.

Resultaten visar också att helhetslösningen kan realisera en förfrågan om vridmoment från föraren på ett tillfredsställande sätt. Slutligen visade även resultaten att justeringar av onlineregleringens parametrar spelar en stor roll för dess beteende.

Nyckelord: Olinjär Modellprediktiv Reglering, Dynamisk Programmering, Optimal Reglering, Optimering, Drivlina, Hybridfordon, Växlade System.

Acknowledgments

We would like to thank the people at CEVT for the great hospitality and giving us the opportunity to perform this thesis. It has been a great learning experience on this very interesting subject.

We would like to give a special thanks to our supervisor Karthik Prasad at CEVT for his encouragement and willingness to assist in any situation. We would also like to thank our examiner Jonas Fredriksson at Chalmers University who has guided us on to the right path throughout the project.

Finally we would like to thank our respective families for the continuous support and encouraging words that has helped us in finishing this thesis.

Daniel Hultgren and Teodor Husmark, Gothenburg, May 2020

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

BSFC	Brake Specific Fuel Consumption
DE	Disconnect Element
DP	Dynamic Programming
GIS	Geographic Information System
ICE	Internal Combustion Engine
NLP	Non-Linear Program
NMPC	Nonlinear Model Predictive Control
OCP	Optimal Control Problem
SoC	State of Charge

Vehicle and Environment Parameters

η_{ICE}	ICE efficiency	%
η_{P1}	Electric motor 1 (P1) efficiency	%
η_{P2}	Electric motor 2 (P2) efficiency	%
ρ_{fuel}	Fuel density	$\mathrm{kg/m^3}$
ρ_{air}	Air density	$\mathrm{kg/m^3}$
A	Vehicle frontal area	$\mathrm{m^2}$
c_{FHV}	Fuel heating value	J/kg
c_{d}	Vehicle drag coefficient	–
c_{rr}	Vehicle roll resistance coefficient	–
E_{bat}	Battery Capacity	J
g	Gravitational acceleration	$\mathrm{m/s^2}$
J_1, J_2	Inertia of shaft 1 and 2	$\mathrm{kg\,m^2}$
J_{m}	Equivalent inertia of the vehicle mass	$\mathrm{kg\,m^2}$
m	Vehicle mass	kg
r_{wheel}	Wheel radius	m
T_{d}	Driver requested torque	N m

Definitions

d_{s}	Sample distance	m
η^{t}	tanh fit of efficiency map	%
$\mathbb{J}$	Cost matrix	–
ω^{opt}, T^{opt}	Optimal operating point	–
ω_1, ω_2	Angular speed of shaft 1 and 2	rad/s
ψ^{E}	Trigger signal	–
θ	Winning margin	%

t_s	Sample time	s
ξ	BSFC	g/kWh
ζ	SoC	%
a	Acceleration	$\mathrm{m/s^2}$
a^max	Maximum acceleration magnitude	$\mathrm{m/s^2}$
E_ICE	Engine events	–
F_air	Air resistance force	N
F_grad	Gradient force	N
F_roll	Roll resistance force	N
i_DE	DE state $\in \{0, 1\}$	–
i_gear	Gear number	–
i_ICE	ICE state $\in \{0, 1\}$	–
i_mode	Mode number $\in \{\mathrm{EM}, \mathrm{Serial}, \mathrm{Parallel}\}$	–
j	Jerk	$\mathrm{m/s^3}$
j^max	Maximum jerk magnitude	$\mathrm{m/s^3}$
m_fuel	Fuel mass	g
N_c	Control horizon length	–
N_p	Prediction horizon length	–
$n_{\hat{a}}$	Number of acceleration comfortability threshold violations	–
$n_{\hat{j}}$	Number of jerk comfortability threshold violations	–
N_gears	Number of gears	–
N_g	Grid size	–
N_modes	Number of hybrid modes = 3	–
P^max	Max rated power	W
r_gear	Total gear ratio	–
T^max	Max rated torque	N m
t^E	Election period	s
T_c	Computational time	s
T_s	Travel time	s
v	Vehicle speed	m/s
V_fuel	Fuel consumption per 10 km	l/10km

1

Introduction

This chapter introduces the issue and relevance of hybrid powertrain control, earlier research and presents the trend of making control predictions based on GPS and route data and why that can be beneficial for control performance. The chapter further states the project aim and limitations along with the main and sub research questions.

Bad air quality, among other motivators, forces emission legislation and regulations to become stricter over time. For instance, the NOx emission limit was reduced by 58 % in relation to the previous year when the Euro 6 regulation for passenger cars was implemented in 2015 [1]. Emission legislation regarding other pollutants such as Particulate Matter (PM) and CO2 are following the same trend which forces a response by vehicle manufacturers. These factors among with the public's general interest of lowering the contribution to global warming has placed hybrid powertrains in the spotlight. By seeking the optimal power split between the power sources, a Hybrid Electric Vehicle (HEV) can in most cases outperform the conventional powertrain in terms of fuel economy and emissions [2]. However, a hybrid powertrain requires advanced and resource consuming control techniques in order to reach its full potential, a field which has not fully been explored yet [3]. This motivates the importance of projects which develops high-end energy management controllers for HEV powertrains to compete with, and eventually replace, the regular and more harmful combustion powertrains.

A hybrid vehicle consists of multiple power sources which allows the power demand by the powertrain to be satisfied in multiple ways. Despite the added complexity of a hybrid powertrain, the driver interface is still similar to the driver interface of a vehicle with a conventional powertrain. The simplicity can be preserved by allowing software to control the power split of the power sources during various driving scenarios. This necessitates the need for clever control logic which decides how much to utilize each power source in every driving scenario. Optimal control is becoming a more frequent control methodology used for this task. Optimal control is a family within control theory which formulates control inputs based on optimality of an objective function. The ability to specify an objective function allows for intuitive tuning using weights while remaining optimal.

The technical progress regarding computing power and algorithmic developments [4] has enabled a switch from the previous rule-based control approach to the optimal control approach within hybrid powertrain control. The movement towards more

powerful control schemes opens the ability of including more features to the control solution, such as predictive behavior. This can come in the form of GPS and route data which allows the controller to predict future road conditions and prepare for them, in order to further optimize the fuel economy and driver experience. Previous studies have shown that including such predictions leads to more sophisticated and intelligent control actions and hence overall control performance [5]. Furthermore, the implementation of predictive ability opens up the possibility of new features such as allowing the driver to specify a desired state-of-charge of the vehicle battery when arriving at the destination.

Making such predictions is computationally heavy and requires a mathematical and often nonlinear model of the powertrain. Thus the computational time increases rapidly with the prediction length and hence eliminates the ideal scenario of being able to predict over an entire driving route in an online manner. The benefit with such predictions is that control actions formulated by the controller in the beginning of the route can be influenced by predictions of the powertrain's future and even terminal state. This would for instance allow the controller to manage the state-of-charge in the most optimal manner during the whole route while making sure that it reaches a specified terminal value at the end of the route.

Since the optimality of the solution increases with the prediction length, researchers have started to investigate if predictions can be based on the full driving route without affecting the computational time of the online controller. A promising approach is to couple an offline optimization step to the online controller. Since the optimization is offline, it does not have to be as fast as for online applications. The offline optimization is usually carried out with Pontryagin's Minimum Principle (PMP) or Dynamic Programming (DP). Its purpose is to predict the behavior of the powertrain over the full driving route and based on these predictions, the optimal reference signals with respect to a specified objective function are formulated. The reference signals are then treated by the online controller. By following the reference signals, the online controller enforces a behavior stated by the offline controller. Hence, the behavior of the online controller is influenced by the long prediction of the offline controller which entails the previously stated benefits of predicting over the full route without affecting the computational complexity of the online controller.

Previous research that has pursued this coupled approach is presented in Section 1.1. From the research within this area it can be noted that the field has not yet been fully explored. A coupled solution requires a combination of an offline optimization approach and an online control scheme. This thesis will therefore investigate a new combination where the offline optimization utilizes the common DP algorithm while the online controller uses the Nonlinear Model Predictive Control (NMPC) scheme. NMPC is considered as a promising candidate to perform the tasks of the online controller. NMPC can formulate control inputs through prediction-based optimizations of an objective function over user defined horizons while satisfying a set of specified input and state constraints. Unlike the most common form of MPC, namely linear MPC, the NMPC scheme can handle both nonlinear dynamics and constraints without excessive simplifications and linearizations which is suitable for powertrain dynamics. Furthermore, by allowing the NMPC to make short predictions on top

of the already prediction-based references it may lead to further optimality and requires investigation. The thesis is therefore aimed towards both the vehicle industry and the academic audience whom are active within the research field of powertrain control.

1.1 Previous research

The results of an experimental benchmark of offline full route optimization and online actuator control for a hybrid powertrain have been presented by Chasse A. and Sciaretta A. in [5]. The offline optimization uses the PMP approach on a sophisticated dynamical model of the powertrain. The online component then uses a Equivalent Cost Minimization Strategy (ECMS) as control scheme which aims at minimizing the energy cost of both fuel and electric energy by converting the consumption figures into an equivalent unit. The results presented emphasizes the benefit of adding a full route offline optimization to the online actuator control. Their choice of including a detailed dynamical model of the powertrain in the offline optimization surely increases the accuracy of the optimization but excluding it from the online controller might also entail some disadvantages. The real control performance and behavior of the powertrain relies heavily on the fact that the environment behaves similarly or close to identical to the conditions of which the offline optimization was carried out. In a real driving scenario, the chances are high of being exposed to unexpected disturbances along the way which causes the driver to act and hence interfere with the offline optimized driving profile. It can therefore be argued that the solution presented by Chasse A. and Sciaretta A. in [5] is limited regarding disturbance robustness. There is therefore a need to expand the research within the offline-online coupled control architecture by investigating if the performance can be enhanced by moving the detailed dynamical models of the powertrain from the offline optimization to the online actuator controller which this project also seeks to investigate.

In addition to the valuable benchmark results of an offline-online coupled optimal control strategy presented by Chasse A. and Sciaretta A. in [5], Yang W. et al. [6] presents the results from an offline velocity optimization strategy for unmanned hybrid mining trucks. By adopting velocity and state-of-charge as state variables, their offline optimization with dynamic programming generates an optimal velocity trajectory which can enhance the fuel economy of their series hybrid mining truck by 26.59 % under the same travel time.

1.2 Problem statement

Multiple attempts at offline-online coupled optimal powertrain control can be found in previous research such as in the previously mentioned article by Chasse A and Sciaretta A (2011) in [5]. However, the field has not fully been explored and requires further investigation. In the literature, the majority of the papers found presents solutions with a sophisticated offline optimization while restraining the online control

component, usually with a rule-based ECMS controller. With algorithmic developments and enough computational power available in the vehicles of today and certainly tomorrow, the prerequisites are in place for more complex online optimal control algorithms to be used. One such promising algorithm is the NMPC approach which formulates control inputs based on optimality of a cost function subject to a predefined set of constraints for a dynamical system of nonlinear character. The main problem for this thesis is therefore to investigate the benefits and disadvantages of coupling an established offline optimization technique in terms of DP with a more complex and advanced NMPC online control. Such investigation leads to the following main research question.

Main Research Question: *What are the strengths and weaknesses of coupling an offline optimization with an online NMPC for hybrid powertrain control in terms of fuel consumption and drivability, compared to using a standalone online NMPC?*

In order to assist answering the main research question, several sub questions are formulated.

How can NMPC be implemented to fit the purposes of hybrid powertrain optimal control?

How can the control solution be adapted to regard driver needs in terms of acting on driver requested torque?

How does the choice of horizon lengths influence the overall behavior of the control solution in terms of driveability and fuel consumption?

1.3 Aim

The aim of this project is to develop an online NMPC controller which manages the energy usage and velocity of a hybrid vehicle with three driving modes and a gearbox while meeting offline optimized references in terms of SoC and velocity, computed using DP. Additionally, the effects of including an offline optimization step to assemble a coupled solution will be evaluated in relation to a solely online control solution. The evaluation will be based on relative fuel consumption and drivability metrics for a given driving route.

1.4 Limitations

The main focus of this thesis is to develop the online controller of the coupled solution. Thus, limited time will be allocated for developing the offline optimization step and only a basic one will be created. The main compromise in the offline optimizer will be to have a fixed gear ratio. This limits the vehicle to only freeway driving and implies that the reference signals which are treated by the online controller are only accurate for such driving conditions. Regarding the development of the online NMPC, dynamic models for tasks such as changing hybrid mode will not be used and will instead be assumed that such actions are instantaneous. The online controller will not be given the feature to apply the mechanical brakes for control

purposes. The online controller will only control longitudinal movement, steering is not involved at all. Additional loads that are not contained by the powertrain, such as AC, heated seats etc, will not be modeled and not regarded by the control solution. Neither the offline optimizer nor the online controller will consider any detailed battery dynamics. Finally, a key element in the online controller is the ability to measure the relevant system states. These measurements will be artificial readings of sensors in the simulation model and will be assumed to be free from the presence of noise and any delays. Hence, filtering approaches to deal with noise removal will not be considered in this project.

1.5 Thesis outline

In the following Chapter 2, the theory in terms of concepts and technical approaches utilized in the thesis are explained in order to prepare the reader for the work method which is described by Chapters 3, 4, 5, 6 and 7. Chapter 3 presents the concept design of the coupled solution and the process of gathering and preprocessing the data required for development and testing. Chapter 4 presents the mathematical modeling of the vehicle and powertrain required for the online controller due to NMPC being a model based control approach. Chapter 5 explains the functionality and development methodology of the offline optimizer denoted C1. Chapter 6 declares the development process of the online NMPC denoted C2. In this chapter, C2 is broken down in to several sub components where the functionality and development of each is presented along with the fully assembled C2. Chapter 7 describes how the system was implemented in order to evaluate its performance. The results acquired from previous chapters are then presented and discussed in Chapter 8. The findings are then compiled and put in relation to the formulated research questions to draw conclusions in Chapter 9. Lastly, Chapter 10 lists a number of future work tasks which can be performed to further enhance the performance of the developed control solution.

2

Theory

This chapter aims to explain various concepts and technologies utilized in the thesis, in order to prepare the reader for the system description in the upcoming chapters.

2.1 Dynamic programming

This section builds on the theory presented in [7]. Dynamic programming can be performed with either forward or backward recursion and affects the definitions of transfer and objective functions among others. This section presents dynamic programming with backwards recursion.

Dynamic programming is an optimization method which can be applied to various optimization problems. The key approach of DP is to break down the original problem into subproblems. The subproblems can then be solved for optimality and the solutions from each subproblem can be reassembled to find the optimal solution of the original problem. Since this solution relies on optimal solutions of subproblems, the DP approach relies of the principle of optimality first defined by Richard Bellman as "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision".

DP is closely related to shortest path problems. However, a big advantage with DP is that the network diagram does not have to be stored explicitly to find the solution since it builds on recursion. Furthermore and as the name implies, the problem has to be of dynamical character in order to solve it with DP. This means that the original problem must be able to be broken down into a number of *stages* and where the movement through the stages is done by sequential decision making. At each stage, the optimal solution is determined by a number of subproblems, each containing only one variable. Each subproblem is defined by a number of potential *states* for the corresponding stage. The optimal decisions, represented by *decision variables*, are found by regarding the potential states in each subproblem. Applying a decision at each state results in a new set of states and hence a new stage where the process can be repeated to find the next optimal decision. Hence, the stages can be connected by optimal decisions to the subproblems which assembles the optimal solution to the original problem.

2.1.1 Key Components of DP

Stage partition: $t = 1, .., T$
Each stage partition is a subproblem of the original problem.

Decision Variables (Input): $u_t, t = 1, .., T$
Influences the behavior of the system. The optimal value of the input is found by regarding the subproblems at each stage and then applied to reach the next stage. The decision variables are usually constrained to a specific set of values as $u_t \in U_t$ at stage t.

State: $s_t, t = 1, .., T$
A state s_t describes an attribute of the system which can include many states. As for the decision variables u, the states s can be constrained within a set of values as $s_t \in S_t$ at stage t.

Model (Transfer function): $s_{t+1} = T_t(s_t, u_t)$
The model connects the state and input for current stage t to the state in the next stage $k + 1$. Some problems may include disturbances w, which usually affects the behavior of the system at each stage t, denoted w_t. For problems which includes disturbances, the model might not yield a value which directly corresponds to a state value s_{t+1}. The model with disturbance can be formulated as $x_{t+1} = T_t(s_t, u_t, w_t)$ where x_{t+1} does not directly match s_{t+1}. In those cases, linear interpolation is often used to find the next state value s_{t+1} that best matches the output of the predictive model x_{t+1}.

Objective function:

- Regular definition:

$$J_t(s_t) = \min_{u_i} \sum_{i=t}^{T} c_i(s_i, u_i)$$

- Recursive definition:

$$J_t(s_t) = \min_{u_t} \{c_t(s_t, u_t) + J_{t+1}(s_{t+1})\}$$

The objective function $J_t(s_t)$ describes the optimal function value computed for the stages up until the t:th stage, $\forall s_t \in S$. The term $c_i(s_i, u_i)$ denotes the cost of applying the input u_i on state s_i at stage i, also known as the cost-to-go.

The recursive definition aims at finding $J_1(s_1)$ which is the cost-to-go from the first to final stage. To start the backward recursion, a boundary condition is required in stage T, which usually initializes the objective function as $J_{T+1}(s_{T+1}) = 0$.. This can be compared to setting the cost of the starting node to zero in a shortest path

problem.

2.1.2 The DP algorithm

1. Set boundary condition, usually as $J_{T+1}(s_{T+1}) = 0$
2. Set current stage as last, $t := T$
3. While $t \geq 1$:
   - $\forall s_t \in S_t$, find u_t which minimizes the recursive definition of the cost function
     $$J_t(s_t) = \min_{u_t}\{c_t(s_t, u_t) + J_{t+1}(s_{t+1})\}$$
   - $t := t - 1$
4. Save the optimal input sequence as $u^* = \{u_1^*, u_2^*, .., u_T^*\}$
5. Do a "forward pass" by applying the optimal input sequence to the predictive model to obtain the optimal state trajectory $s^* = \{s_1^*, s_2^*, .., s_T^*\}$

2.2 CasADi

This project uses the CasADi [8] toolkit in combination with MATLAB to develop the control solution and for other optimization needs. CasADi is an open source MATLAB compatible tool which mainly focuses on numerical optimizations of both linear and nonlinear character. CasADi acts as a framework with components such as symbolic math and helper classes which lets the programmer interact with complicated optimization tools in a familiar way, without the need to worry about compilation or dependencies. The toolkit is shipped with many different optimization solvers, including linear, quadratic and nonlinear solvers. The solver used in this project for the nonlinear programs is the IPOPT[9] library for large-scale nonlinear optimization.

Another reason for choosing CasADi is that it supports generating fast and efficient C-code which can be directly implemented in embedded systems. This is highly beneficial since the ultimate goal is to implement the control solution in the vehicle. However, due to time constraints, this was not further investigated.

2.3 Spatial-derivative state dynamics

In the offline optimization, the reference data (road parameters) of the route is distance based which means that the data points have a known distance between them. This also means that the full route length is known beforehand. However, the time required to traverse the route is a result of the optimization, because it is based on the variable velocity of the vehicle. This creates a problem in the offline optimization since the algorithm requires a fixed horizon length and step size. The result of this is that time-based state derivatives cannot be used since the horizon

length is not fixed in that case. In order to circumvent this issue, spatial-based state derivatives can be utilized instead, as described in [10] and [11]. State equations can be rewritten using the chain rule, as:

$$\dot{y} = \frac{dy}{dt} = \frac{dy}{ds}\frac{ds}{dt} = \frac{dy}{ds}v \Rightarrow \frac{dy}{ds} = \dot{y}\frac{1}{v},\ v \neq 0 \tag{2.1}$$

A special effect appear with this rewrite, namely as $v \to 0^+ \Rightarrow \frac{dy}{ds} \to \infty$. This is intuitive because as the vehicle slows down and eventually comes to a stop, the time required to reach a certain point goes to infinity. Since the offline optimization in this thesis is limited to not reach such low speeds, this problem will not manifest here. However, it is important to remember with future development.

2.4 Model predictive control

Model Predictive Control (MPC) has increased in popularity within recent years since it has become computationally affordable by recent algorithmic developments [4] and hardware advancements. The main benefits of the MPC control technique is its ability to formulate control actions based on real-time optimization of a linear constrained multi variable and quadratic objective function [12]. At each controller time step, the result of the real-time optimization is a sequence of control signals which minimizes the objective function, also referred to as the cost function. The number of control moves in the sequence is called the *control horizon*, denoted N_c. The objective function is not only optimized at the current time but also over a future time segment by predicting the system outputs, the length of the future time segment is called the *prediction horizon* [13], denoted N_p. A special property of the MPC approach is the application of the *receding horizon* control law which implies that only the first input of the acquired sequence is applied to the plant. With knowledge of this concept, the main working scheme of the MPC control approach can be expressed as below [14].

- Find input sequence which minimizes the objective function over the prediction horizon.
- Apply only the first input of the obtained sequence of inputs according to the receding horizon law to the plant.
- Repeat at the next controller time step.

The MPC formulation consists of the objective function and the set of constraints. The constraints also include the state constraints which define the dynamic behavior of the system over time, which gives the predictive behavior. The objective function can be established in many ways, one way is to use a quadratic objective function as:

$$J = \sum_{i=1}^{N_\mathrm{p}} \boldsymbol{x}^{i\top} Q \boldsymbol{x}^i + \sum_{i=1}^{N_\mathrm{p}-1} \boldsymbol{u}^{i\top} R \boldsymbol{u}^i \tag{2.2}$$

Where $\boldsymbol{x}^i$ contains the system states, and $\boldsymbol{u}^i$ denotes the control inputs. Q and R are square matrices containing quadratic costs for the states and inputs respectively. Further note that the objective function can take any form as long as it is quadratic for the regular linear MPC.

The objective function will be minimized by the optimization algorithm, subject to a set of constraints which can be established as:

$$\begin{aligned} \boldsymbol{x}^{i+1} &= f(\boldsymbol{x}^i, \boldsymbol{u}^i) \\ \boldsymbol{u}^i &= \boldsymbol{u}^{N_\mathrm{c}} \ \forall \ i \in (N_\mathrm{c}+1, N_\mathrm{p}) \\ \boldsymbol{x}^i &\in \mathcal{X} \ \forall \ i \in (1, N_\mathrm{p}) \\ \boldsymbol{u}^i &\in \mathcal{U} \ \forall \ i \in (1, N_\mathrm{p}-1) \end{aligned} \tag{2.3}$$

where f is the state integration function, also known as the prediction model, $\mathcal{X}$ is the set of feasible system states and $\mathcal{U}$ is the set of feasible inputs. The second constraint adds the ability to pick a different, shorter control horizon as explained earlier by setting all remaining inputs in the prediction horizon equal to the last free input. Note that since all constraints must be linear for the linear MPC, f must also be linear, which means the system itself must be linear.

2.5 Nonlinear model predictive control

Nonlinear Model Predictive Control (NMPC) calculates the control actions in a similar manner as for regular MPC. However, in the case of NMPC, both the prediction model of the system dynamics and the constraints can be nonlinear. The objective or cost function in the optimization does not have to be quadratic and can instead be linear or nonlinear [4].

NMPC falls under the category of optimal control problems (OCP). An objective function is minimized subject to a set of constraints which can either be of discrete, ODE, DAE or PDE character. The essential strategy to solve an OCP of nonlinear character according to the approach of direct optimal control is to first transform it to a nonlinear program (NLP) and then solve the NLP. Such transformation is often called transcription or discretization. The transcription can be achieved by various approaches and the chances of solving the acquired NLP relies heavily on the choice of transcription approach. Consequently, the best solving method depends on the choice of transcription method [15].

This project uses the direct multiple shooting method as transcription strategy in combination with the Interior Point OPTimizer (IPOPT) as NLP solver. The multiple shooting method can be seen as an extension to the single shooting method which repeatedly simulates the system, given an initial guess, until the constraints are fulfilled and the cost function is minimized resulting in an optimized state trajectory. Zero-order-hold or piecewise linear functions are usually used to approximate the continuous input. The multiple shooting method applies the procedure of single shooting but for smaller segments of the trajectory, which can be expressed as

breaking up the system integration into smaller time intervals. For each segment, the system ODEs are solved numerically. By allowing the length of the segments to become shorter, the relationship between the cost, constraints and decision variables can be considered "arbitrarily linear". By splitting the trajectory into segments, the multiple shooting method can handle complicated OCPs, something which single shooting struggles with [16]. Other benefits with multiple shooting is the ability of being able to simultaneously simulate and optimize, can handle unstable systems and is easy to parallelize [17].

The IPOPT solver can handle both nonlinear and nonconvex constraints and objective functions [9] which is suitable for this project.

2.6 Control of switched systems

Many applications, such as applications within powertrain control, often entails both continuous and discrete dynamics. Hybrid powertrains can include both a gearbox and different multiple driving modes which may add discontinuities to the system dynamics. A system which contains a combination of discrete and continuous elements in the dynamics is often referred to as a *hybrid system* and is usually fairly complex. To increase the chances of successful control of such system, the system can be broken up into several subsystems, allowing each subsystem to exclusively contain continuous dynamics. The discrete elements of the dynamics are then represented by allowing for an external unit to deliver switching signals, governing the process of switching between the subsystems to assemble the original system behavior. The architecture of such system representation is often referred to as a *switched system*, see e.g. [18].

3

Concept Design and Data Preparation

This chapter presents the choice of control design and explains the offline-online coupled control concept. The chapter further explains the methodology to acquire and preprocess the data required for simulation and testing during development of the control solution and the final product.

3.1 Concept design

The overall design of the control solution can in a comprehensive manner be illustrated as in Figure 3.1.

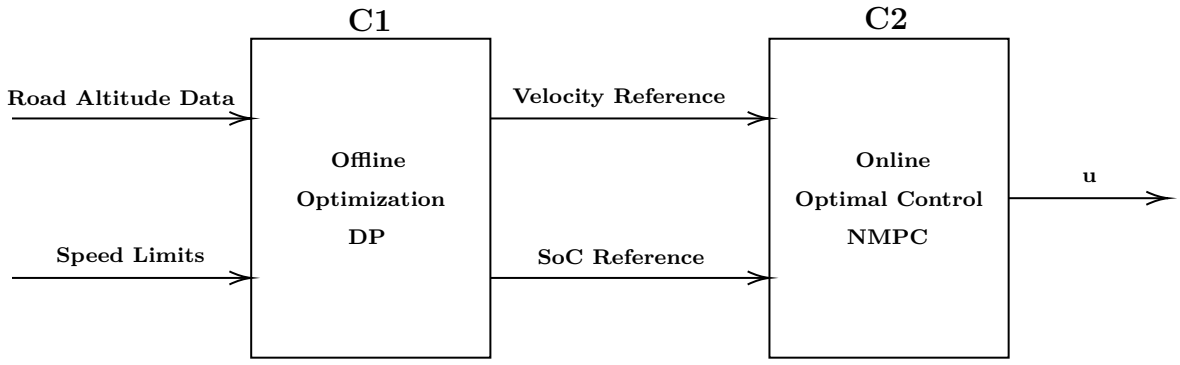


Figure 3.1: A block diagram of the control concept design. The C1 block contains the offline DP optimization which computes references for the online NMPC in the C2 block to track by generating control inputs to the powertrain denoted u.

Road altitude and speed limit data for a full driving route is fed to the offline optimization contained in C1. The C1 optimizer is run before the driving starts and it generates optimized velocity and SoC references which are to be tracked by the online optimal control contained in C2, throughout the driving session. The online optimal controller in C2 generates control inputs to the plant based on the references delivered by C1. The coupling of offline and online optimization is motivated by the belief that the strengths of both methods can be combined into a more optimal controller than if only one of the optimizations are to be used. C1 is able to take the full route into account when optimizing, which it can do by having a relatively large

step size and no hard time restriction. Shorter step sizes are not necessary in the C1 controller since both freeway road inclination and SoC have slow dynamics. The downside of the C1 optimization is that it cannot generate accurate control inputs on a fine resolution since it is run ahead of time and with a large step size, something that is needed for maintaining a responsive and smooth driving experience. This is where the strengths of the C2 controller can be seen. The shorter step size allows it to accurately predict the dynamics of the powertrain and vehicle. Additionally it can respond rapidly to unforeseen events since it utilizes an online control technique. The C2 controller still has a prediction horizon, but it is only in the range of seconds to minutes to make room for a more complex optimization algorithm. A long horizon is still preferable, as that yields a more optimal result, as well as a better predictability. However, since the offline optimized references are generated for the full route, the online control behavior is influenced by information from the full route even if the horizon lengths in the online controller are much shorter.

The choice of using SoC as one of the references was made based on the control objective, which is to minimize the energy consumption subject to an offline optimization. A SoC reference enforces how the battery should be used throughout the route, which is a key aspect for energy management. By offline optimizing a SoC reference curve based on the full length of the route, the behavior of the online controller can be enforced by allowing it to track the reference. Hence, control actions by the online controller will be influenced by knowledge from the full route by following the reference, which is beneficial for the overall energy consumption.

Additionally, previous research has shown that a predictive cruise control has a positive impact on both travel time and energy consumption [19] [20]. One way to achieve the predictive ability is to compute a velocity reference offline which the online controller is then able to follow. Since the controller in this thesis is already built with this methodology, an offline optimized velocity reference can trivially be added to the full solution.

3.2 Route data acquisition

The offline optimization needs to know about road inclination throughout the route in order to be able to compute the SoC and velocity references. The road inclination angle α affects the vehicle both in terms of gravity and friction force, which in turns affects the behavior on the SoC and velocity. Furthermore, the velocity references needs to track the legal speed limit of the road.

This data is available for all Swedish roads for free in the NVDB[1] (National Road Database) after creating an account. Road altitude and speed limit was downloaded as a dataset along a desired route in the GIS (Geographic Information System) data format shapefile. The data was filtered by a specific road number in the download tool on NVDB in order to download only a single stretch of road. The downloaded data however still contain a lot of intersections, side roads and other road features, which is not desirable since the route should be one continuous stretch of road. To

[1] `http://www.nvdb.se/en`

further filter the road data, the freely available program QGIS[2] was used. The route shapefile was loaded in to the program and a shortest path analysis was performed from start to end in order to select a continuous stretch of road. The selection was then inverted and the unnecessary road pieces removed. Finally, the route was manually examined and cleaned for any remaining intersection pieces. The final route was exported to the GIS data format GeoJSON which is a plain text format that can be easily read and interpreted by a MATLAB script.

With the route fully cleaned up, it had to be loaded into MATLAB in an appropriate format for the offline optimizer to utilize. Unfortunately, the data coordinate points in the GeoJSON file is not spatially ordered, they are not ordered in a sequence from route start to route end. Instead, the route is split up into small road segments which are completely unordered in the file. This was solved in MATLAB by starting with the first coordinate point of the route, and then iterating over all coordinates to find the closest one and add that to the end of the route. By repeating this process, the coordinates have been spatially sorted. Then, a moving median filter with horizon length 20 was passed over the data in order to remove outliers in the altitude data. Finally, the data points were resampled with a fixed sample distance as specified by the C1 optimization.

[2] `https://qgis.org/en/site/`

4

Mathematical Modeling

This chapter presents the powertrain architecture and describes the process of modeling the dynamics of the powertrain to fit the purposes of the offline and online components of the control solution.

Since the NMPC control scheme and the DP optimization used in this thesis utilize a model based approach, it is necessary to establish a mathematical model of the powertrain. This is done by finding the appropriate internal states in the system, and deriving continuous state equations for them.

DP uses a simplified version of the mathematical model to simulate the system to formulate the optimal control signals, and thus the optimal reference signals. The NMPC on the other hand uses the detailed dynamical mathematical model presented in this chapter to make predictions of future outputs of the powertrain, which is used to formulate optimal control sequences.

4.1 Vehicle and powertrain modeling

The architecture of the powertrain used in this project can be seen in Figure 4.1.

The powertrain includes three power sources - two electric motors and an internal combustion engine (ICE). The three power sources in the powertrain are connected to the transmission in the specific order to achieve the following three distinct powertrain operating modes - electric mode, serial mode and parallel mode.

In the electric vehicle mode, also denoted EV mode, the propulsion comes from the P2 motor which receives its energy from the battery. The ICE and P1 motor are completely ignored in this mode and the DE is disconnected. The serial mode is similar, however, the ICE drives the P1 motor which acts as a generator and can thus both power the P2 motor and charge the battery. Finally, for the parallel mode, the DE is connected and both the P2 motor and the ICE can contribute to the propulsion. Theoretically, the P1 motor can also contribute to the propulsion of the vehicle, however it is in this powertrain designed primarily as a generator and should thus only be used for that task.

The online controller in C2 uses ICE output shaft speed (ω_1), gearbox input shaft speed (ω_2), State-of-Charge (ζ), and distance (d) as state variables. ω_1 and ω_2 are the anti-derivatives of the shaft accelerations and are thus required for modeling the

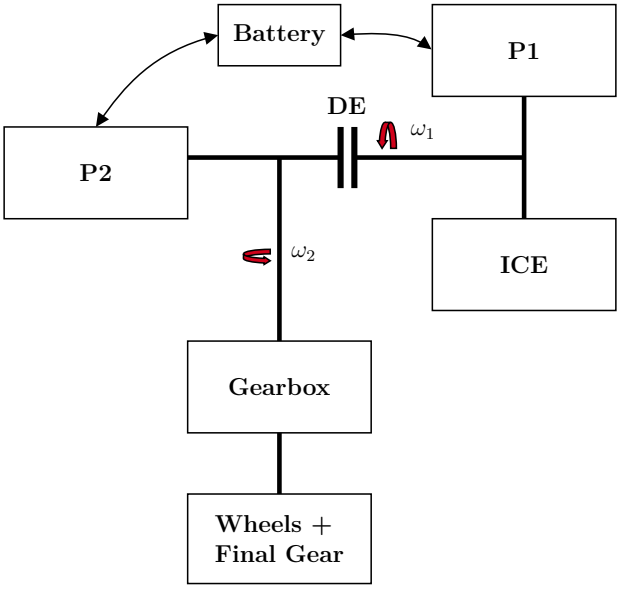


Figure 4.1: The powertrain architecture. DE denotes the disconnect element which manages the coupling between the ω_1 shaft and the rest of the powertrain.

powertrain dynamics. The SoC is necessary for predicting the battery behavior and the influence of the power sources on it. Finally, the distance d is necessary in order to pick the distance-based reference values on the route used for tracking velocity and SoC.

In order to control the powertrain behavior, a set of input signals are available. The DE can be disconnected or connected using the logic signal i_{DE}. The controller is able to request a torque from the different power sources using T_{P1}, T_{P2} and T_{ICE}. It is assumed in this thesis that the motors can always respect the torque request, except for if the signal requests beyond the motors current max torque. Finally, the gear of the gearbox can be set using the integer signal i_{gear}. This gear, along with the final differential gear results in the total gear ratio r_{gear}. It is assumed that the plant responds instantly for these signals.

Furthermore, it is assumed that all four state variables are measured on the plant with no sensor dynamics or noise. To conclude, the inputs and outputs of the powertrain is represented as a system block, seen in Figure 4.2.

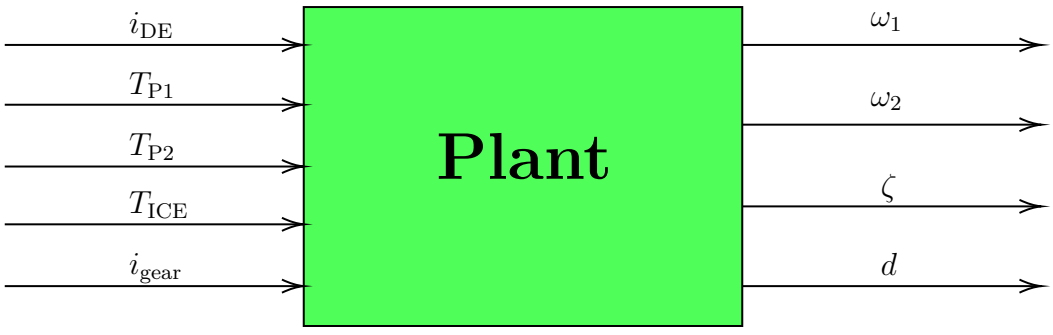


Figure 4.2: Inputs and outputs of the plant.

The DE is modeled as a discrete system with disconnected and connected as the two available states which are represented $i_{\mathrm{DE}} = 0$ and $i_{\mathrm{DE}} = 1$ respectively. The state of the DE causes the powertrain to switch between two distinct discrete modes, which have different dynamics. When the DE is connected, the two shafts get linked and thus the state equations for ω_1 and ω_2 become equal. When DE is disconnected, the two shafts are independent systems, only connected by the battery. The state equations for corresponding connected and disconnected DE are:

$$
\begin{aligned}
&i_{\mathrm{DE}} = 0: \\
&\qquad \dot{\omega}_1 = \frac{1}{J_1}\left(T_{\mathrm{ICE}} + T_{\mathrm{P1}}\right) \\
&\qquad \dot{\omega}_2 = \frac{1}{J_2 + J_{\mathrm{m}}}\left(T_{\mathrm{P2}} - T_{\mathrm{load}}\right) \\
&\qquad \dot{\zeta} = \frac{-T_{\mathrm{P1}}\omega_1\eta_{\mathrm{P1}} - T_{\mathrm{P2}}\omega_2\eta_{\mathrm{P2}}}{E_{\mathrm{bat}}} \\
&\qquad \dot{d} = v
\end{aligned}
\tag{4.1a}
$$

$$
\begin{aligned}
&i_{\mathrm{DE}} = 1: \\
&\qquad \dot{\omega}_1 = \frac{1}{J_1 + J_2 + J_{\mathrm{m}}}\left(T_{\mathrm{ICE}} + T_{\mathrm{P1}} + T_{\mathrm{P2}} - T_{\mathrm{load}}\right) \\
&\qquad \dot{\omega}_2 = \dot{\omega}_1, \ (\omega_2 = \omega_1) \\
&\qquad \dot{\zeta} = \frac{-T_{\mathrm{P1}}\omega_1\eta_{\mathrm{P1}} - T_{\mathrm{P2}}\omega_2\eta_{\mathrm{P2}}}{E_{\mathrm{bat}}} \\
&\qquad \dot{d} = v
\end{aligned}
\tag{4.1b}
$$

The state equations for ω_1 and ω_2 is simply Newton's second law of motion for angular movement. The torques from the power sources are modeled such that a positive torque produces mechanical energy, and a negative torque consumes mechanical energy. This means that a negative torque recharges the battery for the two electric motors, and the vehicle load also acts as a negative torque.

In the state equations, J_{m} is the equivalent inertia of the vehicle mass, which can be calculated using:

$$
J_{\mathrm{m}} = \frac{r_{\mathrm{wheel}}^2 m}{r_{\mathrm{gear}}^2} \tag{4.2}
$$

The SoC is modeled as:

$$
\zeta = \frac{1}{E_{\mathrm{bat}}} \int_0^{T_{\mathrm{s}}} P \mathrm{d}t \tag{4.3}
$$

where P is the sum of the electrical power from the two electric motors. This model does not capture any battery efficiency losses, which can vary with different SoC

levels and power inputs. The model is used despite its limitations in order to reduce the complexity of the optimization problem.

The total load from external forces can be formulated by a load torque on the ω_2 shaft represented by T_{load} in Equation (4.1) and can in forces be expressed as:

$$T_{\text{load}} = \frac{(F_{\text{air}} + F_{\text{roll}} + F_{\text{grad}})\, r_{\text{wheel}}}{r_{\text{gear}}} \tag{4.4}$$

The external vehicle forces can consequently be expressed as:

$$F_{\text{grad}} = m \cdot g \cdot \sin(\alpha) \tag{4.5a}$$

$$F_{\text{roll}} = m \cdot g \cdot c_{\text{rr}} \cdot \cos(\alpha) \tag{4.5b}$$

$$F_{\text{air}} = c_{\text{d}} \cdot A \cdot \rho \cdot \frac{v^2}{2} \tag{4.5c}$$

where g is the gravitational acceleration constant, c_{rr} is the vehicle roll resistance coefficient, c_{d} is the vehicle drag coefficient, A is the vehicle frontal area, ρ is the air density and v is the speed of the vehicle which can be calculated from ω_2 using:

$$v = \frac{\omega_2 r_{\text{wheel}}}{r_{\text{gear}}} \tag{4.6}$$

4.2 Engine modeling

The characteristics of the ICE contained by the powertrain are represented by an efficiency map and a max torque curve. These are key components in the optimization steps since they affect both the components in the cost function and the constraints which the optimization is subject to. Unlike the offline optimization in C1, the optimal controller in C2 has a strict requirement on being computationally fast. This becomes a problem when the NMPC solver has to deal with the raw efficiency map and torque curve of the ICE which contains aggressive derivatives, something which many solvers have trouble dealing with. In order to deal with this problem, the efficiency map can be simplified by using the curve fitting technique. However, one of the components to be minimized in the cost function is the fuel consumption. Hence it is more intuitive to work with a consumption map rather than an efficiency map. The efficiency map can be converted to a consumption map by using:

$$\xi = \frac{1}{\eta_{\text{ICE}} \cdot c_{\text{FHV}}} \tag{4.7}$$

where c_{FHV} is the Fuel Heating Value which defines the amount of heat power released when combusting a specific amount of fuel. By multiplying it with the efficiency, the engine power is achieved.

The obtained consumption map is then curve fitted with a 2D polynomial of degree 2 in the angular velocity dimension and degree 3 in the torque dimension. This reduces the accuracy to some degree but is a small compromise for the vastly quicker solving speed it provides. The curve fitted map retains the same features as the original map which is the main importance for the NMPC solver.

To further increase the speed of the optimization, the curve of maximum engine torque in relation to engine speed is also curve fitted with a polynomial of degree 4.

4.3 Electric motor modeling

The two electric motors P1 and P2 are modeled as power conversion blocks, without any modeling of the internal electric behavior. This is done to simplify the powertrain model and allow us to specify the motor dynamics using only an efficiency map, denoted η. From the efficiency map, the max rated torque $T^{\max}$ and max rated power $P^{\max}$ of the motor can be extracted. Figure 4.3 represents the shape of the torque-speed curve of a generic electric motor. The max rated torque of the motor is the torque value of the max rating curve in the constant torque region. Likewise, the max rated power of the motor can be extracted by picking a point on the max rating curve in the constant power region, and calculating the power as $P = \omega T$. The motors also specify a continuous torque and power rating, which is the maximum torque and power you can request under a sustained load from the motor. When working at an operating point above the continuous rating, the motor housing is not able to dissipate the waste heat quickly enough, which can eventually damage the motor windings or other internal components. Modeling this behavior requires additional states in the system model, thus the motors are limited to only the continuous rating.

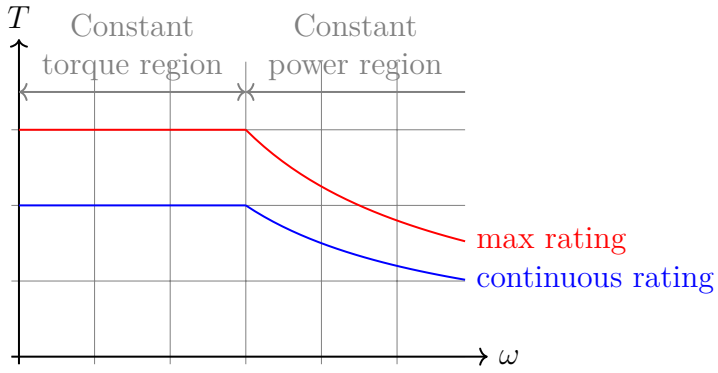


Figure 4.3: The torque-speed curve of a generic electric motor.

As mentioned earlier, the motors are modeled as power conversion blocks. That is, they can be seen as a stateless system block with the formula $P^{\text{out}} = P^{\text{in}} \cdot \eta$. The definition of P^{in} and P^{out} can however change depending on if the requested torque is positive or negative, since a negative torque would turn the motor into a generator instead, thus reversing the direction of the energy flow. For a motor that is producing torque, P^{in} is electrical energy from the other motor or the battery,

and $P^{\text{out}} = T\omega$, and vice-versa for a generating motor that is consuming torque. In order to simplify the state equations, the efficiency has been modified so that it is inverted for positive torque values. The definition of η^{f} is:

$$\eta^{\text{f}}(T) = \begin{cases} \frac{1}{\eta}, & T > 0 \\ \eta, & T < 0 \end{cases} \tag{4.8}$$

The result of this definition is that P^{in} is now always the mechanical energy and P^{out} the electrical energy.

The efficiency map of an electric motor is a function that maps a torque and angular speed to an output efficiency. The angular speed is always defined as positive, but the torque can be both negative and positive. Figure 4.4 shows a cross-section of the efficiency map of a generic electric motor at an angular speed above 0 and the redefinition of it by using Equation (4.8).

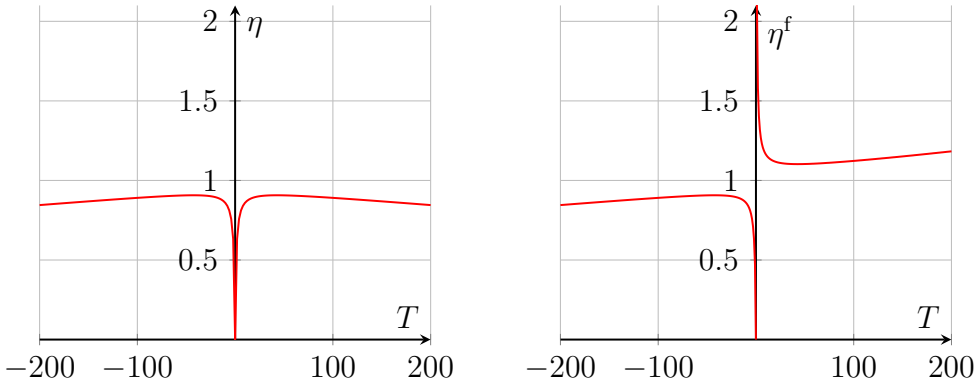


Figure 4.4: The cross-section of a typical efficiency map and the redefinition of it of an electric motor at an angular speed operating point above 0.

At $T = 0$ the energy flow changes direction which leads to a discontinuity in η^{f}. For this case, the function is only piecewise continuous and piecewise differentiable, two factors which many NMPC solvers have trouble dealing with without severe performance penalties. One example why this is an issue is due to the fact that the solvers needs to bring the system forward in terms of calculating the new states, usually done by discrete stepping methods. These methods relies on derivatives of the state trajectories which hence needs to be differentiable over the whole trajectory and where discontinuities may interfere with this property at some trajectory pieces.

A raw efficiency map with the behavior of the typical efficiency map seen in Figure 4.4 was firstly used in the project. However, the discontinuity at $T = 0$ led to a very slow solving speed. Thus the decision was made to redefine and simplify the efficiency map by firstly replacing it with a hyperbolic tangent function, denoted η^{t}, which can be seen in Figure 4.5.

As seen in the figure, the η^{t} function is greatly simplified, but it captures the necessary components of the η^{f} function while remaining continuous and differentiable over the entire domain. One could in theory also consider a straight line fitted to

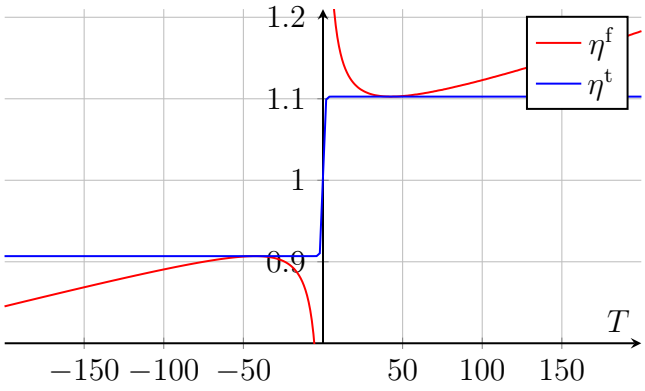


Figure 4.5: A fitted tanh function in blue compared to the discontinuous η^{f}.

the asymptotes of the η^{f} function, which would have an even less complex derivative for the NMPC solver. However, the downside of such a line is that the NMPC solver would believe that torque values close to 0 would have a near perfect efficiency and thus attempt to work in those points more often. This is definitely not true though given the actual efficiency function. The tanh representation solves this problem by being steep in the neighborhood of $T = 0$.

The dependency of angular speed in the efficiency map was decided to be ignored in the η simplification since the shape of it would contribute very little to the decision process in the NMPC solver and would only add unnecessary complexity. The torque dimension is of much more importance since it captures the concept of positive/negative power which is vital for the solution.

4.4 Optimal engine operating point

A major benefit of the serial hybrid mode is the ability to put the ICE at any desired operating point. This is possible because it is not connected to the wheels, so the ω_1 shaft can work at any angular speed. Furthermore, the acceleration of the vehicle is no longer dependent on the torque from the ICE or the P1 motor which means the torque for these two motors can be picked freely as well. The ability to pick any operating point means the engine can constantly operate at its peak efficiency point, which means that it can output as much mechanical power per unit of fuel as requested. However, the constraints of other system components must be kept in mind. Common limiting factors are the torque and power limits of the P1 generator, and the charging power limit of the battery. Additionally, the efficiency of the generator must be regarded as well.

Since this problem contains nonlinearities in the form of the efficiency maps and also a set of constraints, a nonlinear program is built and solved to find the optimal operating point. The formulated NLP is:

$$
\begin{aligned}
\min_{\omega_1, T_{\mathrm{ICE}}} \quad & -\eta_{\mathrm{ICE}}(\omega_1, T_{\mathrm{ICE}})\eta_{\mathrm{P1}}(\omega_1, T_{\mathrm{P1}}) \\
\text{s.t.} \quad & T_{\mathrm{P1}} := -T_{\mathrm{ICE}} \\
& -T_{\mathrm{P1}}^{\max} \leq T_{\mathrm{P1}} < 0 \\
& -P_{\mathrm{P1}}^{\max} \leq T_{\mathrm{P1}}\omega_1 \\
& 0 \leq \omega_1 \leq 5000\,\mathrm{RPM} \\
& -T_{\mathrm{P1}}\omega_1\eta_{\mathrm{P1}}(\omega_1, T_{\mathrm{P1}}) \leq P_{\mathrm{bat}}^{\max}
\end{aligned}
\tag{4.9}
$$

which is solved using CasADi and IPOPT.

The result of solving this NLP is an operating point for the ICE which is able to generate electrical power as efficient as possible but still respecting the system constraints. This operating point is denoted ω_1^{opt} and $T_{\mathrm{ICE}}^{\mathrm{opt}}$. The battery power constraint can be exchanged to an equality constraint in order to specify a specific desired power output if necessary.

A feature of the NMPC controller in C2 is to be able to choose any specific serial charging power during operation. To allow this, it must be able to freely choose either the shaft speed ω_1 or the torque T_{ICE}. We chose to keep the shaft speed fixed, and let the controller tweak the torque since that allows it to instantly adjust the power instead of having to operate the power via the angular acceleration derivative. After minimizing, it was found that the best ω_1 operating point is the same one as for the fixed speed and torque operating point. Thus, that same ω_1^{opt} will be used for this case.

5

Development of C1: Offline Optimized SoC and Velocity References

According to the concept design in Section 3.1, the control solution is divided into two main components, C1 and C2. This chapter presents and explains the development process of the offline optimization component contained by C1.

Using established route data, the offline optimization can be done over the route. The optimization is achieved using the DP algorithm, which is explained in Section 2.1. MATLAB does not have any built-in DP solver, instead a third-party solution is used which is created by O. Sundström and L. Guzzella[21][1]. This DP solver, henceforth referred to as the *dpm* function, is suited for optimal control problems since it is based on objective functions, model equations and infeasiblity matrices.

5.1 Establishing the state-space grid

The model equations for the system are established in a MATLAB function that is supplied to the dpm function. In order to minimize the amount of dimensions required for the DP algorithm, the mathematical model of the powertrain has to be simplified. One such simplification is to only consider the system as a hybrid powertrain with two driving modes, electric and serial, and ignore the parallel driving mode. This means that the propulsion is achieved only using the P2 motor, and the ICE serves as a battery charger that is either running at the computed optimal operating point or turned off. Additionally, the gearbox is fixed at a specific gear ratio. With these simplifications, the system requires only two states, the vehicle speed v (which is proportional to ω_2) and SoC ζ. ω_1 is not necessary since the dynamics between P1 and the ICE is not modeled. Furthermore, the distance d is not needed either since the optimization is performed using spatial derivatives as explained in Section 2.3 (just as a state for the time t is not required in time derivative optimizations). Another benefit with the spatial domain is the distance dependency of the reference signals. With such reference signals, the ability of the online controller to cope with tracking error is better, since the reference changes over distance and not over time; the reference can never de-synchronize which may

[1]Downloadable at `https://idsc.ethz.ch/research-guzzella-onder/downloads.html`

happen with time-based references.

Along the two states, two inputs are required, T_{P2} which controls the torque from the P2 motor and the logic signal i_{ICE} which controls the ICE state. Finally, in order to reduce oscillating behavior, a memory state for the i_{ICE} input is implemented. This state holds the previous i_{ICE} input applied. The combination of the three states and two inputs results in a 5-dimensional problem.

The 5-dimensional space is constructed using the built in functionality of the dpm function, by specifying the number of grid points in each dimension, along with the lower and upper bound. The following grids on each respective dimension are established:

Table 5.1: The established DP grid. N_{g} is the number of grid points. $v_{\max}$ and $v_{\min}$ is the upper and lower bound on the speed limit of the route.

Dim	Name	N_{g}	Lower bound	Upper bound
1	ζ	201	20 %	80 %
2	v	16	$v_{\min} - 5\,\mathrm{km/h}$	$v_{\max} + 5\,\mathrm{km/h}$
3	$i_{\mathrm{ICE}}^{\mathrm{prev}}$	2	0	1
4	T_{P2}	21	$-200\,\mathrm{Nm}$	$200\,\mathrm{Nm}$
5	i_{ICE}	2	0	1

The established grid results in a problem with $201 \cdot 16 \cdot 2 \cdot 21 \cdot 2 = 270144$ elements, which for the dpm algorithm takes roughly 0.4 s per distance step to compute on a modern PC given the supplied model equations. On a 50 km route with a sample distance of 50 m, which is 1000 steps, this results in a total computational time of approximately 7.5 min. A sample distance of 100 m results in 500 steps in the optimization, but with a total computational time of approximately 3.2 min. Hence, the value of the sample distance is a compromise between optimization accuracy and computational time. On freeway driving, the slow dynamics of the environment along with the high speeds means high sample distances are still feasible without instability or inaccuracy problems. Additionally, the Δv cost which will be explained soon has a more desirable impact on higher sample distances as that produce a smoother behavior.

By altering N_{g}, the computational time of the program increases or decreases. A doubled N_{g} in any dimension results in a doubled computational time, likewise a twice as short sample distance also results in a doubled computational time. This means that the problem can quickly become infeasible to compute in reasonable time unless care is taken to optimize N_{g} across the dimensions. The decided N_{g} for each dimension were found through trial and error experiments, where the overall drivability and performance of the simulation result were compared between runs. A high N_{g} for the SoC was determined necessary due to its slow dynamics which means it changes little between the stages and thus finer detail is necessary for the cost. During the simulation runs, it was also noted that both v and T_{P2} did not require as fine detail.

The lower and upper bounds are selected to minimize the distance between the

grid points and thus maximize their use. In order to optimize life-time, a vehicle battery should not be charged/discharged beyond roughly 20-80 %, as explained in [22]. The velocity grid is bounded by the upper and lower speed limit of the road in the route, with some added margin. The P2 motor is able to deliver more torque than the specified bounds, but during normal driving, torques greater than 200 Nm in magnitude are rarely necessary. The lowered bounds allow for a finer resolution on the control input to the P2, so that is more desirable. Finally, i_{ICE} and $i_{\text{ICE}}^{\text{prev}}$ are discrete states, thus 0 and 1 were picked as bounds.

5.2 Integrating the states

The parameterized grid, along with the vehicle parameters and route data, is fed to the dpm function. The function then calls the supplied MATLAB function for every stage where the model equations and objective function are evaluated. The states are updated using the Euler forward integration method with the distance derivatives. In the following equations, d_{s} is the constant sample distance between each step.

Firstly, the velocity state is updated. This is achieved using the torque balance equation:

$$\begin{aligned} \frac{\mathrm{dv}}{\mathrm{ds}}(k) &= \frac{1}{m}\frac{(T_{\text{P2}}(k) - T_{\text{load}}(k))r_{\text{gear}}}{r_{\text{wheel}}}\frac{1}{v(k)} \\ v(k+1) &= v(k) + d_s\frac{\mathrm{dv}}{\mathrm{ds}}(k) \end{aligned} \tag{5.1}$$

Secondly, the SoC state is updated. This is done by calculating the power consumed by the P2 motor and the power produced by the ICE, through the P1 generator. P_{P1} and P_{P2} are defined as the output power of each motor, thus efficiency losses must be accounted for. The SoC derivative is the sum of the powers, and is then updated in the same way as v:

$$\begin{aligned} P_{\text{P1}}(k) &= i_{\text{ICE}}(k) \cdot T_{\text{ICE}}^{\text{opt}} \cdot w_{\text{ICE}}^{\text{opt}} \cdot \eta_{\text{P1}}^{\text{t}} \\ P_{\text{P2}}(k) &= T_{\text{P2}}(k) \cdot \frac{v(k) r_{\text{gear}}}{r_{\text{wheel}}} \cdot \eta_{\text{P2}}^{\text{t}} \\ \frac{\mathrm{d}\zeta}{\mathrm{ds}}(k) &= \frac{P_{\text{P1}}(k) - P_{\text{P2}}(k)}{E_{\text{bat}}}\frac{1}{v(k)} \\ \zeta(k+1) &= \zeta(k) + d_{\text{s}}\frac{\mathrm{d}\zeta}{\mathrm{ds}}(k) \end{aligned} \tag{5.2}$$

where η^{t} is used here since it is also used in the C2 controller which should be able to follow this reference as closely as possible.

5.3 Defining the objective function

For each DP stage, the objective function is evaluated according to:

$$J = K_{\text{fuel}} P_{\text{P1}} d_{\text{s}} + K_{v_{\text{err}}} v_{\text{err}}^2 + K_{\Delta v} \Delta v^2 + K_{\Delta i_{\text{ICE}}} \Delta i_{\text{ICE}}^2 \tag{5.3}$$

where:

$$\begin{aligned} v_{err} &= d_{\text{s}} \left(v(k+1) - v_{\text{ref}}(k+1) \right) \\ \Delta v &= \frac{1}{d_{\text{s}}} \left(v(k+1) - v(k) \right) \\ \Delta i_{\text{ICE}} &= \frac{1}{d_{\text{s}}} \left(i_{\text{ICE}}(k) - i_{\text{ICE}}^{\text{prev}}(k) \right) \end{aligned}$$

The $P_{\text{P1}} d_{\text{s}}$ term represents the "fuel cost" which is used to minimize ICE usage and thus fuel consumption. v_{err} is used to make the vehicle track the speed limit on the route. The Δv cost is necessary to smooth the velocity changes for better drivability, however the benefit of this cost is slightly compromised due to the limited resolution of the fixed grid. That is, since the algorithm has a finite discrete set of what inputs it can pick, it can only achieve a finite discrete set of Δv values, which may not always be the most desired. Finally, the Δi_{ICE} cost displays the purpose of the $i_{\text{ICE}}^{\text{prev}}$ state. Due to the simple way that the SoC is modeled, any specific SoC value is just as optimal as any other. This means that the SoC can take many different equally optimal paths through the optimization, which allows the ICE to switch on or off at any time. This can lead to an oscillating behavior which reduces the drivability of the vehicle. By introducing a high cost on switching the ICE on or off, this problem can be eliminated.

5.4 Computing infeasible states

The final calculation of each DP stage is to compute the infeasibility matrix for the dpm solver. The infeasibility matrix contains boolean values for every element in each dimension that indicate whether that combination of states are feasible or not. This allows us to put constraints on both states and inputs:

$$\begin{aligned} 20\,\% \leq \zeta(k+1) \leq 80\,\% \\ -P_{\text{bat}}^{\max} \leq P_{\text{charge}} \leq P_{\text{bat}}^{\max} \\ -0.9 P_{\text{P1}}^{\max} \leq P_{\text{P1}}^{\text{in}} \leq 0.9 P_{\text{P1}}^{\max} \\ -0.9 P_{\text{P2}}^{\max} \leq P_{\text{P2}}^{\text{in}} \leq 0.9 P_{\text{P2}}^{\max} \end{aligned} \tag{5.4}$$

where

$$\begin{aligned} P_{\text{charge}} &= P_{\text{P1}}(k) - P_{\text{P2}}(k) \\ P_{\text{P1}}^{\text{in}} &= \frac{P_{\text{P1}}(k)}{\eta_{\text{P1}}^{\text{t}}} \\ P_{\text{P2}}^{\text{in}} &= \frac{P_{\text{P2}}(k)}{\eta_{\text{P2}}^{\text{t}}} \end{aligned}$$

The first constraint is trivial, the optimizer should prevent charging and discharging the battery beyond the set limits. The second constraint on P_{charge} prevents the difference between P_{P1} and P_{P2} from exceeding the battery charge and discharge capacity which stems from the battery chemistry and thermal constraints. The practical implications of this constraint involves that the ICE sometimes has to be switched on or off in order to fulfill the constraint. In scenarios which involves steep downhills where the vehicle can perform regenerative braking, either the ICE must be turned off or the vehicle must be allowed to speed up in order to prevent charging the battery with too much power. Likewise, in steep uphills, the ICE might need to be turned on in order to prevent discharging the battery with too much power. The final two constraints limits the input power to each motor. The input power is defined as the requested power by the sources, thus the efficiency component is removed since the efficiency is the conversion factor from input to output power. Additionally, the power limits are scaled with a factor of 0.9 in order to prevent the optimizer from picking points that are on the limit, which impedes the C2 controller from making more optimal decisions. A main motivator for using a two-stage structure of the control solution is that the online controller in C2 can further enhance the offline optimized behavior from C1 by allowing some freedom to deviate from the reference signals. Without scaling down the maximum power limits, there is a chance that the C1 optimizer decides to run at maximum power for some segments. In such cases, the C2 controller has limited freedom and is forced to follow the suggested operating point by C1 to follow the reference. By lowering the power limit, the C2 controller can choose operating points both above and below the C1 operating point at all times, thus giving it more freedom.

Beyond the specified infeasibility matrix, lower and upper bound constraints can be set on the terminal values of all states in the dpm framework. This function is used in order to make the SoC meet the desired terminal value.

5.5 Optimization results

With the stage function setup, the dpm solver can be run. It outputs a set of chosen signals over the optimization horizon. The optimized velocity and SoC is saved as vectors where each element is the value separated by d_{s} m apart. During the online optimization, the saved vectors can be loaded and the velocity and SoC reference values are picked using linear interpolation.

6

Development of C2: Online NMPC Controller

This chapter presents and explains the development process of the online NMPC controller contained by C2

For development purposes, C2 is split into additional subcomponents. The methodology presented in this chapter relies on the approach where the functionality of each component is developed while black boxing the other systems. Many iterations are then performed back and forth between the components until the demonstrated product was completed. The following sections aims to explain the various components in C2, how they interact with each other, what sort of problems arose and how they were solved.

C2 uses the Nonlinear Model Predictive Control (NMPC) approach. The control actions are based on minimizing a cost function subject to a predefined set of constraints and plant output predictions which builds on detailed dynamics of the powertrain.

6.1 Online control architecture

With the multiple speed fixed ratio transmission and the three available driving modes, the powertrain can operate in various combinations of driving mode and selected gear. NMPC entails many advantages such as influencing control actions with predictions and the ability to consider constraints in the optimization. However, handling discrete elements such as driving mode switches is a weakness of the NMPC framework and places high demands on the solver which has to treat discontinuities. With such motivation, the proposed control architecture uses a set of NMPC controllers which all have different dynamics depending on the discrete state of the system. The number of NMPC controllers are as many as there are driving mode and gear combinations. By using a set of NMPC controllers, the discrete dynamics of switching driving mode and/or gear can be moved outside of the NMPCs which lets them focus solely on the continuous dynamics of the current driving mode and gear. The processes of selecting the appropriate NMPC controller based on several conditions is dedicated to a component here denoted the *Elector*. The actual control of the vehicle is performed by the *Controller*.

Each of the NMPCs contains a version of the state dynamics presented in Equation (4.1). For each controller, the dynamics are adapted to conform with the specific gear and driving mode. In addition to state of the DE, the presence of the power sources differs between the three driving modes. Hence, the state dynamics for the Electric, Serial and Parallel driving modes can in a more precise manner be written as:

EV:

$$\begin{aligned} \dot{\omega}_2 &= \frac{1}{J_2 + J_\mathrm{m}} \left(T_\mathrm{P2} - T_\mathrm{load}\right) \\ \dot{\zeta} &= \frac{-T_\mathrm{P2}\omega_2\eta_\mathrm{P2}^\mathrm{t}}{E_\mathrm{bat}} \\ \dot{d} &= v \end{aligned} \tag{6.1a}$$

Serial:

$$\begin{aligned} \dot{\omega}_2 &= \frac{1}{J_2 + J_\mathrm{m}} \left(T_\mathrm{P2} - T_\mathrm{load}\right) \\ \dot{\zeta} &= \frac{-T_\mathrm{P1}\omega_1^\mathrm{opt}\eta_\mathrm{P1}^\mathrm{t} - T_\mathrm{P2}\omega_2\eta_\mathrm{P2}^\mathrm{t}}{E_\mathrm{bat}} \\ \dot{d} &= v \end{aligned} \tag{6.1b}$$

Parallel:

$$\begin{aligned} \dot{\omega} &= \frac{1}{J_1 + J_2 + J_\mathrm{m}} \left(T_\mathrm{ICE} + T_\mathrm{P1} + T_\mathrm{P2} - T_\mathrm{load}\right) \\ \dot{\zeta} &= \frac{-T_\mathrm{P1}\omega\eta_\mathrm{P1}^\mathrm{t} - T_\mathrm{P2}\omega\eta_\mathrm{P2}^\mathrm{t}}{E_\mathrm{bat}} \\ \dot{d} &= v \end{aligned} \tag{6.1c}$$

Each NMPC contains the corresponding dynamics presented in these equations. The state equations all include T_load which in turn includes the total gear ratio r_gear as previously presented in Equation (4.4). In order to assist the controller in the serial mode, the ω_1 shaft speed is assumed to always be at its optimal point, thus, its state dynamics does not need to be modeled. In the EV mode, the ICE and P1 motors are completely ignored, thus the ω_1 shaft speed is not necessary here either. Finally, due to DE being connected in the parallel mode, the two states ω_1 and ω_2 are combined into just ω.

The combination of three hybrid modes and the available gears results in multiple NMPCs where each controller contains the state equations for the specific driving mode, adjusted for the specific gear. The optimization objective in terms of the cost function is however identical for all NMPCs:

$$J = K_{\text{fuel}}||\boldsymbol{m}_{\text{fuel}}||_2^2 + K_{V_{\text{err}}}||\boldsymbol{v} - v_{\text{ref}}||_2^2 + K_{\zeta_{\text{err}}}||\,|\boldsymbol{\zeta} - \zeta_{\text{ref}}|\,||_3^3 + K_{\Delta\omega_2}||\Delta\boldsymbol{\omega}_2||_2^2$$

where:

$$\begin{aligned} \boldsymbol{m}_{\text{fuel}} &= \left\{ t_{\text{s}} \frac{\xi^j \omega_1^j T_{\text{ICE}}^j}{1000 \cdot 60 \cdot 60} \;\middle|\; j \in (2, N_{\text{p}}) \right\} \\ \boldsymbol{v} &= \left\{ v^j \;\middle|\; j \in (2, N_{\text{p}}) \right\} \\ \boldsymbol{\zeta} &= \left\{ \zeta^j \;\middle|\; j \in (2, N_{\text{p}}) \right\} \\ \Delta\boldsymbol{\omega}_2 &= \left\{ \frac{1}{t_{\text{s}}}(\omega_2^{j+1} - \omega_2^j) \;\middle|\; j \in [1, N_{\text{p}} - 1]\,,\ \omega_2^1 = x_2(k) \right\} \end{aligned} \tag{6.2}$$

The fuel component is the criteria which makes the optimizer try to minimize the fuel consumption. The V_{err} and ζ_{err} are used to make the controller follow the reference. Finally, $\Delta\omega_2$ acts to smooth the velocity of the vehicle, which increases drivability.

It is important that all the NMPCs share the same cost function, as will be explained later in the Elector sections.

As an additional clarification for Equation (6.2), K_{fuel}, $K_{V_{\text{err}}}$, $K_{\zeta_{\text{err}}}$, $K_{\Delta\omega_2}$ are scalar weights for the corresponding components in the cost function. Furthermore, the fuel consumption over the horizon, $\boldsymbol{m}_{\text{fuel}}$ uses the curve fitted consumption map presented in Section 4.2 which delivers the BSFC at every time instant t, denoted ξ^t. Somewhat unconventionally, the SoC error uses a cubed L3-norm. It was found during testing that if only a squared L2-norm is used, the fuel cost can sometimes be more expensive than the SoC error if the system is far away from the SoC reference. This would make the system fall into a bad state where it will never attempt to recharge the lost SoC since it is cheaper to drain SoC than consume fuel. Having a cubed error makes the cost grow faster which mitigates this issue. Finally, $\Delta\omega_2$ is the numerically differentiated derivative of ω_2. It is differentiated using the two-point method.

The NMPCs applies the direct optimal control approach to convert the OCP to an NLP. Each NMPC uses the multiple shooting approach with an explicit Euler integration scheme on short time intervals to transcribe the OCP to an NLP. The acquired NLP is then solved with the IPOPT solver. The optimization in each NMPC is carried out with respect to a defined set of constraints tied to corresponding driving mode:

EV:

$$\begin{aligned} -T_{\text{P2}}^{\max} \leq T_{\text{P2}} \leq T_{\text{P2}}^{\max} \\ -P_{\text{P2}}^{\max} \leq T_{\text{P2}}\omega_2 \leq P_{\text{P2}}^{\max} \\ -P_{\text{bat}}^{\max} \leq P_{\text{charge}} \leq P_{\text{bat}}^{\max} \\ 0 \leq \omega_2 \end{aligned} \tag{6.3a}$$

where:

$$P_{\text{charge}} = -T_{\text{P2}}\omega_2\eta_{\text{P2}}^{\text{t}}$$

Serial:

$$\begin{aligned} -T_{\mathrm{P1}}^{\max} &\leq T_{\mathrm{P1}} \leq 0 \\ -P_{\mathrm{P1}}^{\max} &\leq T_{\mathrm{P1}}\omega_1^{\mathrm{opt}} \\ -T_{\mathrm{P2}}^{\max} &\leq T_{\mathrm{P2}} \leq T_{\mathrm{P2}}^{\max} \\ -P_{\mathrm{P2}}^{\max} &\leq T_{\mathrm{P2}}\omega_2 \leq P_{\mathrm{P2}}^{\max} \\ -P_{\mathrm{bat}}^{\max} &\leq P_{\mathrm{charge}} \leq P_{\mathrm{bat}}^{\max} \\ 0 &\leq \omega_2 \end{aligned} \tag{6.3b}$$

where:

$$P_{\mathrm{charge}} = -T_{\mathrm{P1}}\omega_1^{\mathrm{opt}}\eta_{\mathrm{P1}}^{\mathrm{t}} - T_{\mathrm{P2}}\omega_2\eta_{\mathrm{P2}}^{\mathrm{t}}$$

Parallel:

$$\begin{aligned} -T_{\mathrm{P1}}^{\max} &\leq T_{\mathrm{P1}} \leq 0 \\ -P_{\mathrm{P1}}^{\max} &\leq T_{\mathrm{P1}}\omega \\ -T_{\mathrm{P2}}^{\max} &\leq T_{\mathrm{P2}} \leq T_{\mathrm{P2}}^{\max} \\ -P_{\mathrm{P2}}^{\max} &\leq T_{\mathrm{P2}}\omega \leq P_{\mathrm{P2}}^{\max} \\ 25 &\leq T_{\mathrm{ICE}} \leq T_{\mathrm{ICE}}^{\max} \\ \omega &\leq \omega_{\mathrm{ICE}}^{\max} \\ T_{\mathrm{ICE}} &\leq T_{\mathrm{load}} - T_{\mathrm{P1}} \\ -P_{\mathrm{bat}}^{\max} &\leq P_{\mathrm{charge}} \leq P_{\mathrm{bat}}^{\max} \end{aligned} \tag{6.3c}$$

where:

$$P_{\mathrm{charge}} = -T_{\mathrm{P1}}\omega\eta_{\mathrm{P1}}^{\mathrm{t}} - T_{\mathrm{P2}}\omega\eta_{\mathrm{P2}}^{\mathrm{t}}$$

Most constraints are self-explanatory. The constraints pertaining to the different $T^{\max}$ and $P^{\max}$ are necessary in order to not request more torque or power from the motors, battery and engine than what is available and can physically be delivered. Note that $T_{\mathrm{ICE}}^{\max}$ here is the curve-fitted max torque curve. The parallel mode has an arbitrary lower torque limit on the ICE of 25 Nm. The friction losses in the ICE is modeled by the engine efficiency, which is only present in the $\boldsymbol{m}_{\mathrm{fuel}}$ equation and not defined separately in the state equations. This means there is an inherent loss of being in parallel mode which the controller does not know about. Without this constraint, the controller would often pick $T_{\mathrm{ICE}} = 0$ which it would assume results in no fuel consumption and no friction losses, but actually results in the engine idling.

Finally, the $T_{\mathrm{ICE}} \leq T_{\mathrm{load}} - T_{\mathrm{P1}}$ constraint implies that the torque generated from the ICE must not be larger than what is necessary to oppose the load torque and the generated P1 torque. This is used to prevent the P2 motor from consuming any torque delivered by the ICE, and is necessary since the P2 motor is designed primarily to act as a torque producing motor in this powertrain, not a generator. Note that no constraints on SoC nor velocity is included since it is only supposed to follow the reference values. Additional unnecessary constraints only lead to more complex computations for the solver and increases the risk of infeasibility which is undesired.

6.2 Elector and Controller architecture and functionality

The online powertrain controller is split up in two parts, the Elector and the Controller, as mentioned earlier. The separation of the control algorithm is done in accordance with the switched system control scheme. The purpose of the Elector is to orchestrate the Controller by computing which hybrid mode and gear is the most appropriate for the current driving scenario. The purpose of the Controller is to compute the input signals to the plant.

Both components contain NMPC solvers with a high computational cost, but having a split up system lets them run asynchronously and in parallel which is ideal for the real time operating systems in vehicles. Furthermore, the Elector does not need to be run as often as the Controller since the current vehicle state scenario changes with a long time constant when driving on the freeway. The Controller however, needs to run more often in order to produce a smooth and responsive driving behavior.

6.2.1 Elector

In order to decide which NMPC to use for controlling the vehicle among the set of NMPCs, a decision process must be held. This is performed by the Elector, which holds an election process at a fixed time period. The election process consists of polling all feasible NMPCs, "asking" them which is the best to use at the current situation given the current measured states. Simplified, the Elector elects the NMPC which is able to achieve the lowest value of the objective function until the next election is held.

The block diagram representation of the Elector can be seen in Figure 6.1. The Elector takes the current plant state and gear as input, and outputs which NMPC to use for control by specifying the desired mode and gear, denoted $i_{\text{gear}}^{\text{elected}}$ and $i_{\text{mode}}^{\text{elected}}$. The distinction between i_{gear} and $i_{\text{gear}}^{\text{elected}}$ is important since they have different sample times, i_{gear} is which gear the powertrain currently uses, while $i_{\text{gear}}^{\text{elected}}$ is the gear that the powertrain should use the next controller timestep.

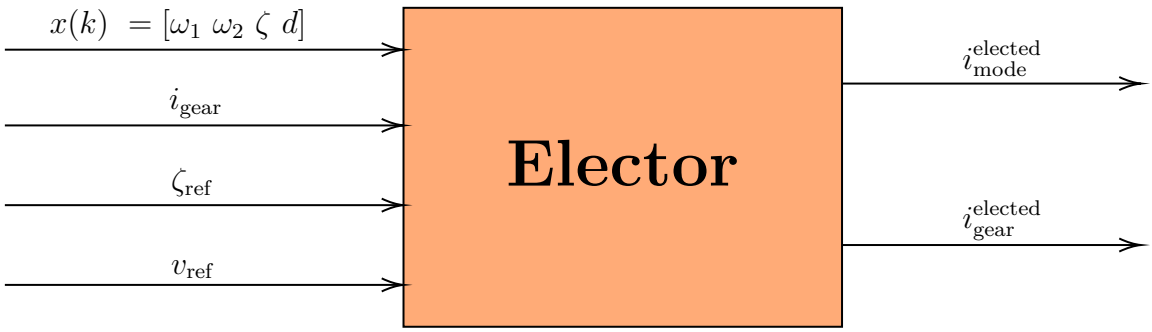


Figure 6.1: System model of the Elector.

The election process starts by screening the set of NMPCs for those who are able to provide a feasible solution. The aspect which influences the chances of a feasible

solution the most is the gear selection. Dependent on the vehicle speed, some gear selections are infeasible since they would violate the constraints on the angular speeds of ω_2. For the parallel driving mode, the vehicle speed also affects the angular speed of ω_1 since the DE is connected. Both shafts in the power train are set to have an upper speed limit of 5000 RPM to represent mechanical limits. The upper RPM limit of ω_1 is further motivated to prevent the possibility of exceeding the redline RPM limit of the engine. Additionally, the ICE is set to have a lower limit on the angular speed of the output shaft ω_1 to prevent stalling the engine, set at 1000 RPM.

With the NMPC screening complete, each NMPC in the feasible set delivers the optimal value of their objective function by predicting the plant outputs based on the control actions in accordance to the MPC control scheme. In order for the feasible NMPCs to deliver the correct optimal objective value, they are supplied with the measured current plant states. Additionally, since ω_2 has different values for a specific vehicle speed v based on which gear is in use, the measured ω_2 must be adjusted to the respective NMPCs gear. Otherwise, the prediction in the NMPC optimization will be incorrect. The optimal objective function values are stored in a cost matrix $\mathbb{J}^{N_{\text{modes}} \times N_{\text{gears}}}$. The cost of the infeasible NMPCs are represented as ∞ in the cost matrix.

With the cost matrix $\mathbb{J}$ established, the decision process can be held. This can be seen as an additional optimization problem where the most optimal cost in the cost matrix is to be picked given a set of criterion. Advanced control methods can be utilized for complex decision strategies, however in this project a simple rule-based hysteresis controller is employed since it aligns well with the scope of this project. Advanced decision controllers can for more sophisticated control solutions be attractive since the effectiveness of such strategies scales well for decision problems of higher complexity as if auxiliary loads were to be considered for instance.

6.2.2 Elector hysteresis

The hysteresis element of the rule-based controller is implemented by introducing the concept of winning margin. The "winner NMPC" is the NMPC with the lowest cost in the cost matrix. However, for the winner NMPC to be elected, it must pass the winning hysteresis criteria:

$$\mathbb{J}_{u,v} < \theta \cdot \mathbb{J}_{i,j} \tag{6.4}$$

In the equation, (u, v) is the winner NMPC mode and gear, and (i, j) is the previously elected NMPC mode and gear. Finally, $0 < \theta \leq 1$ is the *winning margin*.

By setting $\theta = 1$, no hysteresis will be taken place. Instead, by setting $\theta = 0.9$ for example, the winner NMPC must have a 10 % better cost than the previously elected NMPC. If the winner NMPC does not fulfill the winning criteria the NMPC which currently controls the plant will be re-elected to control the plant until the next election is held.

The inclusion of a hysteresis element in the rule-based controller is implemented to increase the drivability aspect of the full control solution by reducing the number of driving mode switches through the tunable winning margin θ.

By implementing the hysteresis with only a multiplier criteria has some flaws. By doing so, the hysteresis becomes sensitive to the magnitudes of the objective costs. One could for instance argue that an objective cost of 10^5 is far better than a cost of 10^6 while an objective cost of 8 is only marginally better than a cost of 10 for the same objective function weights. Both cases fulfills a winning criteria of 10 %, but only in the first example would a switch produce any noticeable improvement in vehicle performance. Thus, some better hysteresis criteria could be implemented, however this has not be investigated in this thesis.

6.2.3 Election period and horizon lengths

The Elector periodically holds an election where the fixed time between the elections is determined by the election period t^{E}. When t^{E} seconds have passed, the election process is initiated by a trigger signal denoted ψ^E. The NMPCs then delivers the objective costs by using control and prediction horizons $N_{\mathrm{c}}^{\mathrm{E}}$ and $N_{\mathrm{p}}^{\mathrm{E}}$ respectively, which are solely used within the election process. In order to simplify the tuning of the controller, the control horizon is set to be equal to the prediction horizon, i.e. $N_{\mathrm{c}}^{\mathrm{E}} = N_{\mathrm{p}}^{\mathrm{E}}$. A critical tuning parameter is the relation between the election period t^{E} and the prediction horizon $N_{\mathrm{p}}^{\mathrm{e}}$. The horizon were picked such as $N_{\mathrm{p}}^{\mathrm{E}} = \lceil t^{\mathrm{E}}/t_{\mathrm{s}} \rceil$, where t_{s} is the Controller time step. This results in a prediction horizon that predicts the vehicle behavior up until the next election and thus results in objective costs which are the most relevant for the election process.

The election period t^{E} is a tunable parameter which greatly influences the behavior of the full control solution. By selecting a long election period, the frequency of gear and mode switches is restricted since they can only occur once every election. This implies that the NMPC controlling the plant might not be optimal at all points during control. Since $N_{\mathrm{p}}^{\mathrm{E}} = \lceil t^{\mathrm{E}}/t_{\mathrm{s}} \rceil$, another disadvantage with a long election period is the computational burden that long prediction horizons have.

Computational burden is also something which very short election period can suffer from since the frequency of the elections becomes high. And since $N_{\mathrm{p}}^{\mathrm{E}} = \lceil t^{\mathrm{E}}/t_{\mathrm{s}} \rceil$, using short election periods also reduces the predictive ability of the NMPCs and hence the purpose of using a predictive control approach. Additionally, a shorter election period can have a negative impact on drivability due to rapid NMPC switches if the hysteresis is not tuned well.

In theory, the election period should be selected to represent the speed at which the environment changes, with some limitations. For example, for freeway driving, the road slope changes relatively slowly and there are rarely any interruptions. In city driving however, start-stopping is frequent and perhaps a fixed election period is not a viable option. In this project where we only consider freeway driving the election period should hence be adjusted accordingly.

6.2.4 Controller

The gear and mode decided by the Elector is sent to the Controller, which gets translated to a specific NMPC to use for controlling the plant. At each Controller time step t_s, the Controller receives the current measured plant states and then runs the NMPC for the given inputs. A block diagram with the inputs and outputs of the Controller can be seen in Figure 6.2.

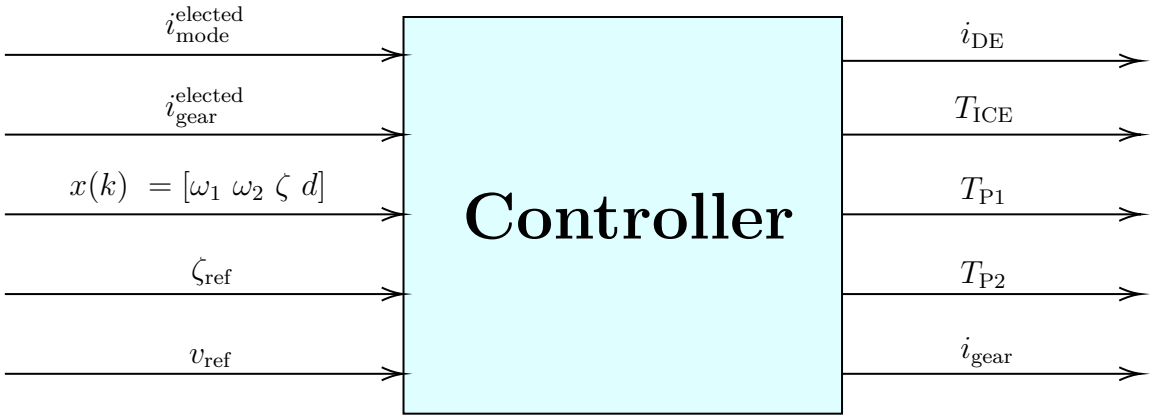


Figure 6.2: Overview of the Controller.

The controller utilizes a time step set to $t_s = 1\,\text{s}$ which can be considered to be fairly large. However, running the NMPCs is a heavy operation and by setting time step to 1 s has a positive impact on the overall computational time. Furthermore, the main purpose of the online controller is to perform well on freeway driving where fast changes are not as frequently occurring as for e.g. city driving which further implies that a time step of 1 s is sufficient.

Similar to the Elector, the Controller has to account for the fact that different gears entails different angular speeds of ω_2. Hence, the Controller checks if a change of gear has been made since the last run. If so, the angular speed of ω_2 is updated with respect to the new choice of gear.

The Controller contains a new independent set of NMPCs used to control the plant. In contrast to the Elector, the NMPCs in the Controller uses different lengths of prediction and control horizons denoted N_{p}^{C} and N_{c}^{C} respectively. As for the Elector, the prediction and control horizons for the Controller are set to be of equal length, i.e. $N_{\text{p}}^{\text{C}} = N_{\text{c}}^{\text{C}}$. The Controller allows for longer horizons lengths than the Elector since only one NMPC is allowed to run in the Controller, something which should be utilized. Using longer horizon lengths in the Controller enhances the predictive performance, since the active Controller can "see further" which implies a smoother control behavior due to increased optimality of the objective function. Smoother control behavior also enhances the drivability aspect since it reduces oscillations and drastic changes of the actuator signals in terms of requested torque to the torque sources of the powertrain.

Since ω_1 is fixed at the optimal operating point in the serial NMPC, the NMPC will not take into account the acceleration of the ω_1 shaft. That is, the torques sent by the

NMPC will not make sure that the shaft holds the specified operating point. Thus, a separate controller is necessary to solve this control problem. Since the dynamics of the ω_1 shaft are relatively simple in the serial mode, this control problem is solved by a basic P-controller which adds an additional signal on top of the T_{P1} and T_{ICE} sent from the NMPC. By modifying the requested torque signals, the P-controller can make ω_1 follow the desired reference. Once the reference is reached and the error is 0, T_{P1} will be equal to T_{ICE} which keeps the angular acceleration at 0. The T_{P1} and T_{ICE} computations in the P-controller are defined as:

$$\begin{aligned} T_{\mathrm{P1}} &:= T_{\mathrm{P1}} + 2(\omega_1^{opt} - \omega_1) \\ T_{\mathrm{ICE}} &:= T_{\mathrm{ICE}} + 2(\omega_1^{opt} - \omega_1) \end{aligned} \tag{6.5}$$

By adding the new signal on top of the computed ones lets the controller still respect the requested torque from the NMPC and also eliminates any steady-state error. Note that this is the desired torque values, in reality the P1 motor and ICE has torque limits, thus the signals will get saturated. The P-controller is run with a sample time of 10 ms in order to produce a fast response. An additional benefit of this controller is that the P1 motor will act as a starter motor for the engine. The value of the P constant is not very sensitive and was found through trial and error via simulations.

6.3 Responding to driver requested torque

Driver interaction is something which the online controller has to consider in order for it to be viable in practice. The type of driver interaction which has been considered in this project is the possibility of the driver to request a specific torque, T_{d}, by pressing the gas pedal for situations such as overtakes. The solution assumes that T_{d} is delivered from the driver and has already been translated from gas pedal angle.

A sudden torque demand from the driver such as in the case of an overtake has to be treated as quickly as possible by the Elector and then delivered by the Controller. A new election is therefore forced by utilizing the trigger signal ψ^{E} which disregards the election period and forces a new election when a driver requested torque is detected. The trigger signal then becomes active when the numerical derivative of T_{d} exceeds a specified threshold, i.e. when the driver suddenly request a large torque. The logic for the trigger signal is:

$$\psi^E = \begin{cases} 1 & \text{if } \left|\frac{\Delta T_{\mathrm{d}}}{\Delta t}\right| > 20\,\mathrm{Nm} \\ 0 & \text{Otherwise} \end{cases} \tag{6.6}$$

When the new election is triggered, the Elector checks if $T_{\mathrm{d}} > T_{\mathrm{load}}$. If so, the driver wants to accelerate beyond the optimized vehicle speed, which the controller has to allow. The main control objective for the NMPCs is therefore to follow the torque reference from the driver and will in the driver engagement mode utilize a modified

cost function. Hence a new set of NMPCs are used, where the cost function in all NMPCs is replaced by:

$$
\begin{aligned}
&i_{\mathrm{DE}} = 0: \\
&\quad J = K_{\mathrm{fuel}} ||\boldsymbol{m}_{\mathrm{fuel}}||_2^2 + K_{\zeta_{\mathrm{err}}} ||\boldsymbol{\zeta} - \zeta_{\mathrm{ref}}||_3^3 \\
&\qquad + K_{T_{\mathrm{d}}} ||T_{\mathrm{P2}} - T_{\mathrm{d}}||_2^2
\end{aligned} \tag{6.7a}
$$

$$
\begin{aligned}
&i_{\mathrm{DE}} = 1: \\
&\quad J = K_{\mathrm{fuel}} ||\boldsymbol{m}_{\mathrm{fuel}}||_2^2 + K_{\zeta_{\mathrm{err}}} ||\boldsymbol{\zeta} - \zeta_{\mathrm{ref}}||_3^3 \\
&\qquad + K_{T_{\mathrm{d}}} || \left(T_{\mathrm{ICE}} + T_{\mathrm{P1}} + T_{\mathrm{P2}}\right) - T_{\mathrm{d}}||_2^2
\end{aligned} \tag{6.7b}
$$

This cost function depends on the state of the DE. If the DE is disconnected, the only power source which provides traction force is the P2 motor. Otherwise if the DE is connected, all power sources can affect the vehicle acceleration. T_{d} is assumed to be constant over the entire horizon.

The other components of the original cost function in Equation (6.2), such as velocity tracking, are omitted because their sole purpose would be to actively impede the requested torque from the driver.

In addition the cost function, another property which differs in the driver engagement mode is the length of the election period. As for regular control, the relations between the horizon lengths and the election period are however kept the same. I.e. $N_{\mathrm{p}}^{\mathrm{E}} = \lceil t^{\mathrm{E}}/t_{\mathrm{s}} \rceil$, $N_{\mathrm{p}}^{\mathrm{E}} = N_{\mathrm{c}}^{\mathrm{E}}$ and $N_{\mathrm{p}}^{\mathrm{C}} = N_{\mathrm{c}}^{\mathrm{C}}$. The election period for the driver engagement mode is set to $t^{\mathrm{E}} = 2\,\mathrm{s}$ and $N_{\mathrm{p}}^{\mathrm{C}} = N_{\mathrm{c}}^{\mathrm{C}} = 5$. By doing so, the election frequency increases and the horizon lengths are short which is suitable since driver interaction yields an unpredictable behavior in general.

The election procedure is then identical to the regular election process. The NMPC which delivers the lowest objective cost and satisfies the winning criteria is elected as the winner NMPC and allowed to control the plant. As for the Elector, the Controller block contains an additional set of NMPCs used for driver engagement mode with the objective cost in (6.7). In order the use the new set of NMPCs, the Controller checks in the same manner as for the Elector if $T_{\mathrm{d}} > T_{\mathrm{load}}$. T_{load} can be computed from the system states so no additional block input is needed.

Responding to driver requested torque is a challenging task since the response has to be fast. The throttle response is a vital aspect for customers. The switched system architecture used in this project makes responding to driver requested torque possible, but comes with a downside. When the driver requests a large enough torque to trigger a new election, both the Elector and Controller switches to the new set of NMPCs with adapted objective costs. This procedure is fairly fast, but the Elector still has to hold an election process to find the NMPC which satisfies the requested torque in the most optimal way. Such election is not instantaneous and has a negative impact on responsiveness.

6.4 Full control solution

By coupling the Elector together with the Controller, the full control system in C2 is achieved. The Elector runs every t^{E} seconds by a timer pulse, or from a change in requested torque. Once the election process has finished, the decided hybrid mode-gear combination is sent to the Controller which is run at every time step to compute the different input torques necessary for the vehicle to follow the SoC and velocity references. The full system solution can be seen in Figure 6.3.

6.5 Evaluation and measures taken to increase drivability

Drivability is the "subjective perception of dynamic performance and comfort for passenger car in response to driver input during longitudinal driving" according to W. Zhou et al.[23]. It is a key aspect when designing vehicle controllers since the comfortability of the driver is of high importance. Drivability is a challenging aspect to evaluate, since it is often a subjective measure. However, some metrics can be still be calculated for a simulation that allows for a rough comparison between runs. The drivability metrics utilized in this report are the number of engine events E_{ICE} and the maximum magnitudes of acceleration $a_{\max}$ [m/s^2] and jerk $j_{\max}$ [m/s^3] over a driven route. An engine event is considered as either starting the ICE or turning it off. In addition to regarding the maximum magnitudes of these metrics, it is also beneficial to regard the occurrences of acceleration and jerk values over specified comfortability thresholds. The acceleration and jerk thresholds are set in accordance with what is presented in [24] to $\hat{a} = 1.47\,\mathrm{m/s^2}$ and $\hat{j} = 0.9\,\mathrm{m/s^3}$. The number of accelerations and jerks which exceeds these thresholds will be counted and represented by $n_{\hat{a}}$ and $n_{\hat{j}}$ respectively. Engine events are important to reduce since starting the engine is a loud process and often causes annoying vibrations for the passengers. This subjective evaluation of drivability builds on the opinion that lower values of the listed metrics increases drivability.

Throughout the project, drivability has always been kept in mind when designing components. The focus has been on reducing the number of mode and gear switches since they often implicate jerky and/or unresponsive behavior. Smoothing control signals and states have also been important when designing elements, which for example explains the extra cost on $\Delta\omega_2$ in the NMPC control scheme.

A benefit with the developed solution is that it contains multiple tunable parameters which affects drivability such as winning margin, election period and horizon lengths. The current state of the solution employs rule-based decisions by the Elector. To further enhance the drivability aspect of the solution, the Elector can be extended to utilize optimization techniques where drivability metrics can be included and hence affect the decisions.

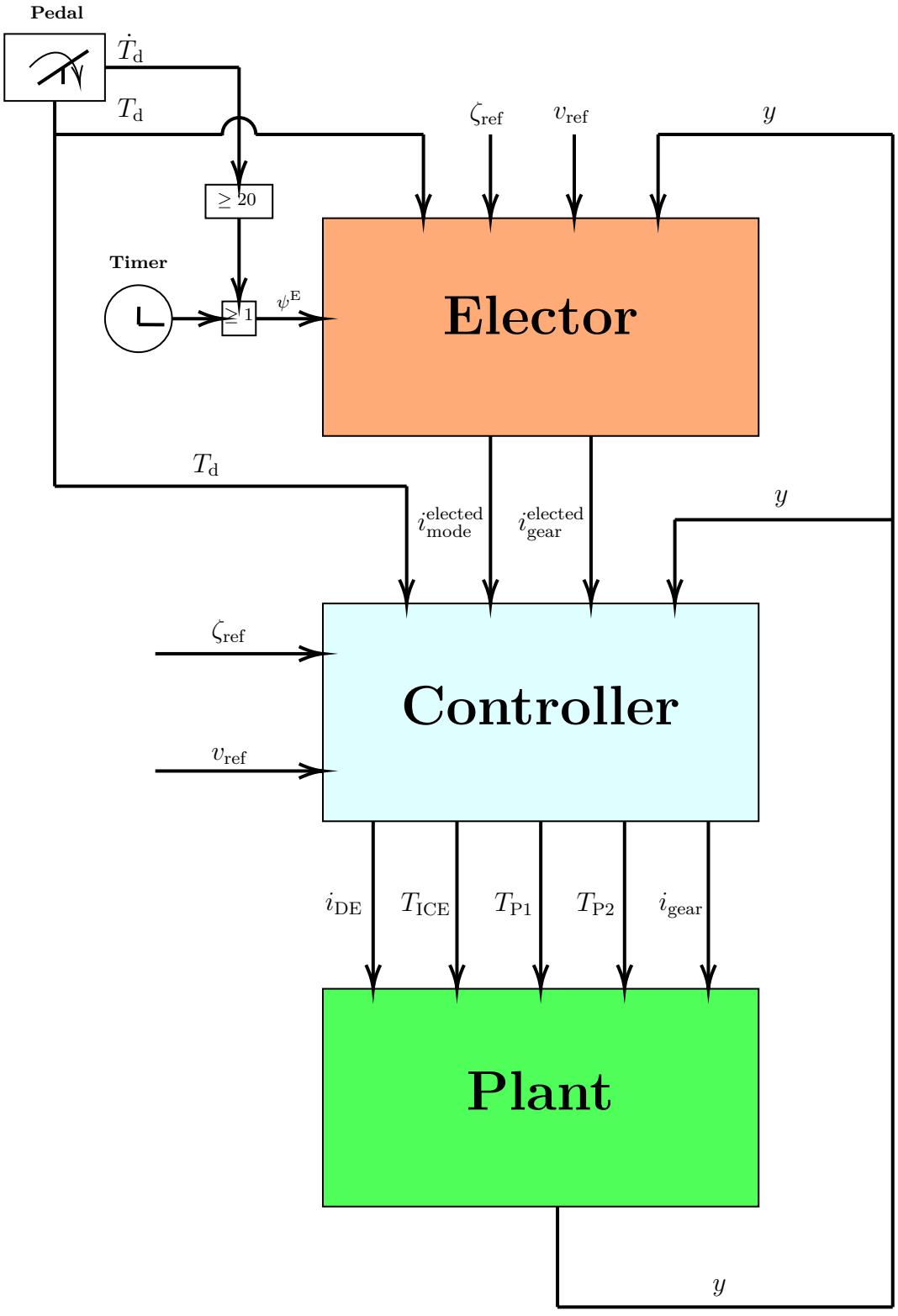


Figure 6.3: Full solution architecture

7

System Evaluation and Simulation

This chapter presents the simulation environment and the framework for evaluating the system.

In order to evaluate the performance of the system and allow for quick iterations between trial and error when tuning, a simulation environment was created using Simulink and MATLAB. A detailed vehicle plant was provided by CEVT that uses the Simscape Driveline add-on for Simulink, which aims to mimic the three mode hybrid powertrain. Simscape Driveline lets you intuitively connect various vehicle and powertrain components such as engines, gearboxes and inertias using block diagrams. The vehicle plant acts as a black box in which the designed controller sends the different input signals and acts on the output signals.

A variable step length was chosen for the simulation, which allows Simulink to adjust the step length when necessary in order to preserve simulation accuracy, while still be relatively fast to simulate. The Controller block has a fixed time step of t_s seconds since that is what the Controller calculations are based on.

The vehicle battery is simulated using only an integrator block which takes P_{charge} that is based on the real efficiency maps of the electric motors as input, and outputs the battery SoC. This is a highly simplified battery model, but suits the scope of this project well.

The ICE is modeled using the Simscape Driveline Generic Engine block, which takes the desired torque as input. Furthermore, supplying the fuel consumption map to the block makes it also calculate the real fuel consumption, which is the one used for metric analysis later. The ICE block does not simulate any engine lag or other dynamics.

The electric motors are modeled using two ideal torque sources, with the saturated desired torque as input. The efficiency loss from the motors are modeled in the battery plant as said before.

All simulations used in this report were performed with an initial SoC of 50 % and a desired terminal SoC of 50 %. Furthermore, all simulations assume that the initial velocity is equal to the reference velocity at distance $d = 0$.

7.1 Controller and Elector implementation

The Controller and Elector consists of two separate MATLAB System blocks which contains code for running the NMPC solvers when desired. The NMPC solvers are implemented as individual independent objects wrapped around CasADi opti functions. The NMPC objects contains an internal state which remembers the NMPC solution of the previous run. By saving the previous solution, it can be utilized in the next NLP solve to 'pre-warm' the solver, which significantly speeds up the computational performance. Additionally, this pre-warming proved highly beneficial beyond just computational time. Due to the non-linearity of the controllers, there can be multiple local minima in the solution. If the solver is not pre-warmed and has to start from scratch, it randomizes the internal variables and optimizes from there. This means it can sometimes, randomly, reach a different local minimum compared to the last solution. The solution at this local minimum can have vastly different optimal variables compared to other minima, thus the controller output can oscillate violently between iterations. By pre-warming, the controller will reach the same local minimum most of the times and thus these oscillations are prevented.

7.2 SoC loss compensation

In order to be able to compare the fuel consumption in a less biased way in between the simulations, it is important that the terminal SoC is equal to the initial SoC as the chosen simulation cases suggests. This is difficult to manage in the simulation since it is not a hard constraint of any controller in the system to achieve this behavior. When computing the final fuel consumption of the run, the result must be compensated to account for any loss or gain in SoC over the run. The battery energy which was "stolen" from the battery can in relation to the energy capacity of the battery, E_{bat}, be expressed as:

$$E_{\text{stolen}} = E_{\text{bat}}(\zeta(0) - \zeta(T_{\text{s}}))[\text{J}] \tag{7.1}$$

Where $\zeta(0)$ and $\zeta(T_{\text{s}})$ is the initial and terminal SoC, respectively. The "stolen" energy can be used as:

$$\begin{aligned} t_{\text{recharge}} &= \frac{E_{\text{stolen}}}{\omega_1^{\text{opt}} T_{\text{ICE}}^{\text{opt}} \eta_{\text{P1}}^{\text{t}}}[\text{s}] \\ m_{\text{fuel}}^{\text{extra}} &= t_{\text{recharge}} \frac{\omega_1^{\text{opt}} T_{\text{ICE}}^{\text{opt}} \xi^{\text{opt}}}{1000 \cdot 60 \cdot 60}[\text{g}] \end{aligned} \tag{7.2}$$

in order to calculate a fuel consumption compensation.

In Equation (7.2), ξ^{opt} is the brake specific fuel consumption [g/kWh] at the optimal serial operating point as described in Section 4.4. The first equation gives the engine running time in serial mode needed to recharge the missing energy. The second equation calculates the mass of fuel required to run the engine at its optimal point for that amount of time. The total fuel consumption could then be computed as:

$$m_{\text{fuel}} = \frac{1}{1000 \cdot 60 \cdot 60} \int_0^{T_s} \omega_1(t) T_{\text{ICE}}(t) \xi(t) \text{dt} + m_{\text{fuel}}^{\text{extra}}[\text{g}] \tag{7.3}$$

where the fraction of $\frac{1}{1000 \cdot 60 \cdot 60}$ is used to convert the fuel consumption from g/kWh to g/Ws. Further note that the integration is performed by the Generic engine block in Simulink.

To obtain the fuel consumption in l/10km, the mass of consumed fuel can be divided by the product of fuel density ρ_{fuel} in grams per liter and the driven distance d in meters as:

$$V_{\text{fuel}} = \frac{m_{\text{fuel}}}{\rho_{\text{fuel}} \cdot d \cdot 10^{-4}} [\text{l/10km}] \tag{7.4}$$

7.3 Route

The majority of the simulations are performed on a route consisting of the first 50 km of National Road 40 from Gothenburg to Borås in Sweden, referred to as the test route. This stretch of road starts with a speed limit of 70 km/h and gradually increases to 110 km/h, which aligns with the scope of the project to only consider freeway driving. Furthermore, the road is a freeway with no intersections which means stops does not have to be taken into account in an ideal scenario.

In order to further evaluate the performance of the solution, an additional and artificial route was created, denoted the "bowl route". This route is only 20 km long compared to the length of the test route which is 50 km. The bowl route was created by replicating a segment of a sine wave. It is simple to its character and only contains one downhill followed by a single uphill to reach the starting altitude. The bowl route consists of a constant speed limit set to 70 km/h.

The characteristics of the two routes can be seen in Figure 7.1.

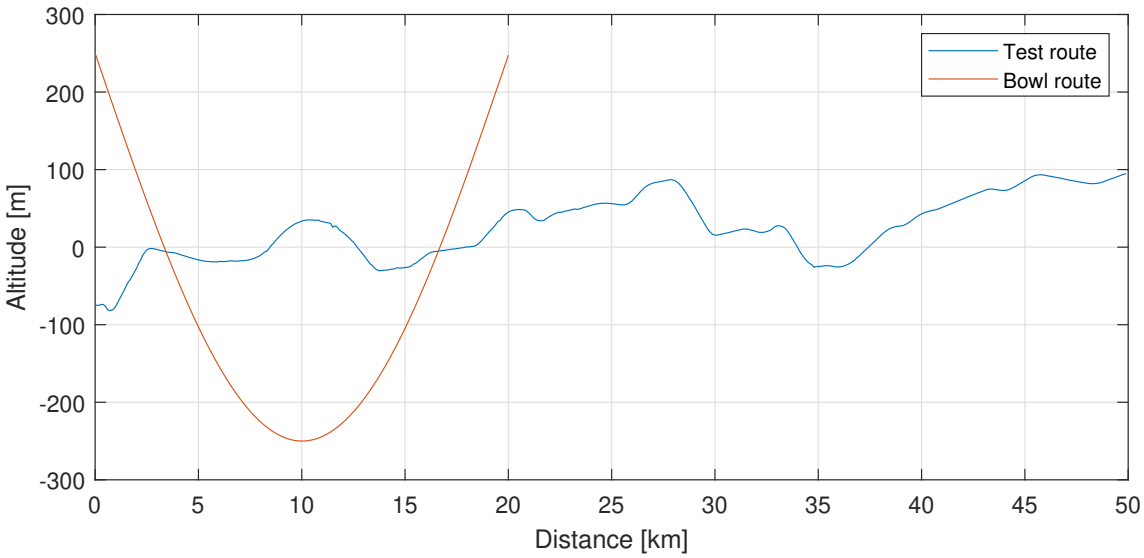


Figure 7.1: Comparison of altitude characteristics for the test and bowl route used to generate simulation results of the developed solution.

8

Results and Discussion

This chapter presents and discusses the project results. In order to present the behavior of the developed control solution, an array of simulations was performed that aims to showcase the solution and lay the foundation for answering the formulated research questions.

8.1 C1: Offline optimization

The DP optimization was performed on the test route with a number of different sample distances in order to find the best sample distance. Figure 8.1 and Table 8.1 shows a comparison of the simulation results between a set of sample distances, $d_s = \{20, 50, 100, 200, 300\}$. All the different optimizations used the same cost tuning for an unbiased comparison:

$$\begin{aligned} K_{\text{fuel}} &= 1 \times 10^{-3} \\ K_{v_{\text{err}}} &= 6 \times 10^{-2} \\ K_{\Delta v} &= 5 \times 10^{6} \\ K_{\Delta i_{\text{ICE}}} &= 1 \times 10^{9} \end{aligned} \tag{8.1}$$

Table 8.1: Optimization metrics for a set of sample distances d_s. E_{ICE} denotes the number of engine events, $v_{\text{err}}^{\max}$ states the maximum deviation from the route speed limits, $a^{\max}$ and $j^{\max}$ is the maximum acceleration and jerk respectively and T_c is the computational time for the specific setup.

d_s	T_s	V_{fuel} [l/10km]	E_{ICE}	$a^{\max}$ [m/s^2]	$j^{\max}$ [m/s^3]	T_c
20	29m 27s	0.7821	7	1.062	2.221	17m 45s
50	29m 13s	0.7731	5	0.643	0.439	8m 20s
100	29m 13s	0.7720	1	0.444	0.081	4m 39s
200	29m 7s	0.7696	2	0.595	0.069	2m 35s
300	29m 11s	0.7660	2	0.411	0.031	1m 36s

As evident in Table 8.1, the different sample distances show little change in fuel consumption. Since the differences in fuel consumption and travel time are small, the comparison was expanded with additional metrics as can be seen in the table.

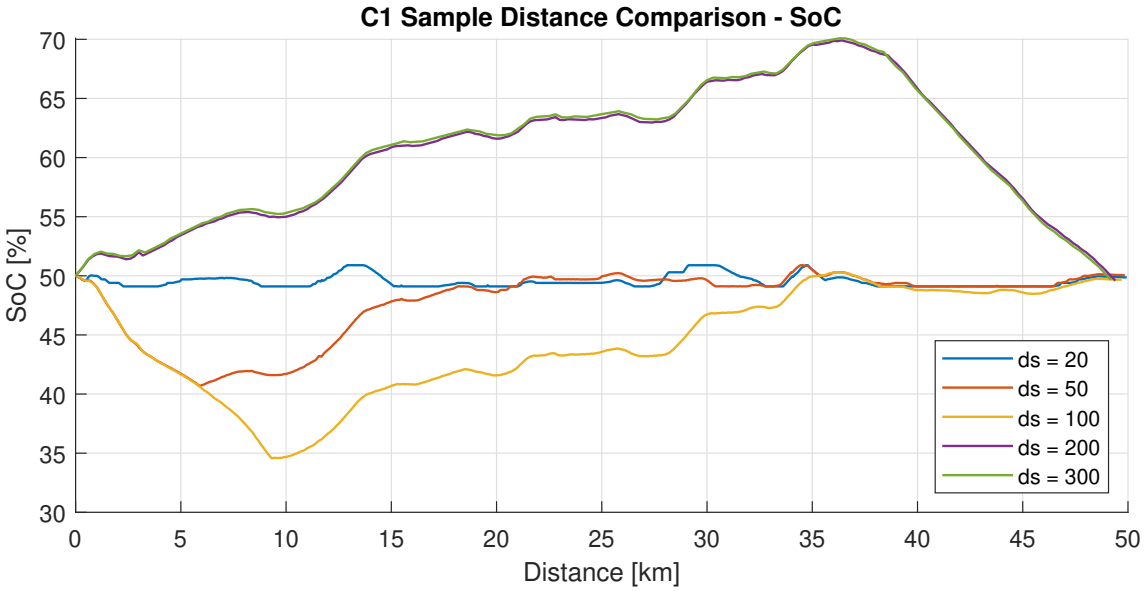


Figure 8.1: SoC reference for a set of different sample distances

By regarding the additional metrics, it is evident that a longer sample distance d_s leads to a smoother behavior of the references. By increasing d_s the number of engine events E_{ICE}, maximum acceleration a_{max} and jerk j_{max} all decreases which is beneficial when regarding the drivability aspect.

Additionally, from the a^{max} and j^{max} metrics in Table 8.1 it can be seen that the smoothness of the velocity reference increases with the sample distance as predicted in Section 5.1. Furthermore, the drivability suffers from lower sample distances as can be seen by the increased number of engine events. Cases with a very low sample distance, such as $d_s = 20\,\text{s}$, may require some re-tuning of the costs in order to achieve similar performance as for the cases with longer sample distances. A theory as to why the number of engine events increase with shorter sample distance is due to the added d_s multiplier in the P_{P1} cost term mentioned in Section 5.3. This makes sense because it would be a larger commitment to turn on the engine for cases with longer sample distance since that would consume more fuel. This could make the optimization more careful with switching it on. The difference in the amount of engine events manifests as the difference in the shape of the SoC reference as seen in Figure 8.1. This is due to the fact that more engine events enables the SoC reference to track the terminal goal SoC of 50 % in an earlier stage.

The different shapes of the SoC trajectory can be explained by referring back to what was mentioned in Section 5.3. Due to the simple model of the battery, any SoC trajectory that starts and ends at 50 % is just as valid as any other. Thus, it does not matter if the SoC increases then decreases, or vice-versa, when considering the control objective.

Picking a best sample distance is difficult since for $d_s \geq 100$ the metrics are mostly the same. In our case, the computational time T_c is not a factor of concern since we have plenty of time to perform the offline optimization. However, for a normal use case it would be a significant factor. Due to this, the sample distance was chosen

as $d_s = 100\,\text{m}$ since it has a good fuel consumption and among the best drivability considering the singular engine event, however it might as well have been picked as 200 m or 300 m without any significant difference regarding their affect on the C2 behavior.

As evident from the figures, the offline DP algorithm is able to achieve a stable and optimized trajectory despite only utilizing a simple Euler forward stepping method. Allowing for more computational complexity, we believe that this method of performing the offline optimization is definitely a viable one.

Early attempts were made where an NMPC solver would perform the offline optimization. However, the combination of large route distances and relatively short sample distances resulted in an extreme amount of prediction horizon steps, which together with the different system states resulted in a completely infeasible problem to solve in a reasonable time. This allows the less computationally heavy DP algorithm to outshine an NMPC solution under these circumstances. However, the drawbacks of the DP algorithm must be kept in mind when developing the algorithm. The most notorious problem we faced is the lack of taking past and future states into account in the cost function, which is possible for NMPC. The oscillating behavior explained in Section 5.3 was solved by introducing the $i_{\text{ICE}}^{\text{prev}}$ state, however the introduction of that results in a doubled computational time, and that is only for a 1-step history.

8.2 C2: Optimal control

In order to evaluate the performance of the online controller, a baseline controller was tuned as good as possible, which is then used to compare to different tunings and simulation cases.

8.2.1 Establishing the baseline controller

With respect to the effects of the election period length presented in Section 6.2.3, the election period used for simulation on the test route was moderately selected as $t^{\text{E}} = 20\,\text{s}$. Likewise, the winning margin was moderately set to 10 % ($\theta = 0.9$). The objective cost was tuned accordingly:

$$\begin{aligned} K_{\text{fuel}} &= 1 \times 10^2 \\ K_{V_{\text{err}}} &= 3 \times 10^2 \\ K_{\zeta_{\text{err}}} &= 1 \times 10^9 \\ K_{\Delta\omega_2} &= 5 \times 10^2 \end{aligned} \tag{8.2}$$

The results from simulations on the test route are presented in Figures 8.2, 8.3 and 8.4. In the plots, the x-axis is color coded to facilitate the reader with determining what driving mode is active under at what times during the simulation. As can be seen for all Figures, the color coding indicates that only EV and Parallel Mode

are used throughout the simulation, which is determined reasonable due to the high speed profile. Serial is less useful in power demanding situations such as freeway driving since the generated energy will be spent immediately anyway, which results in unnecessary energy losses in the electric motors and the battery.

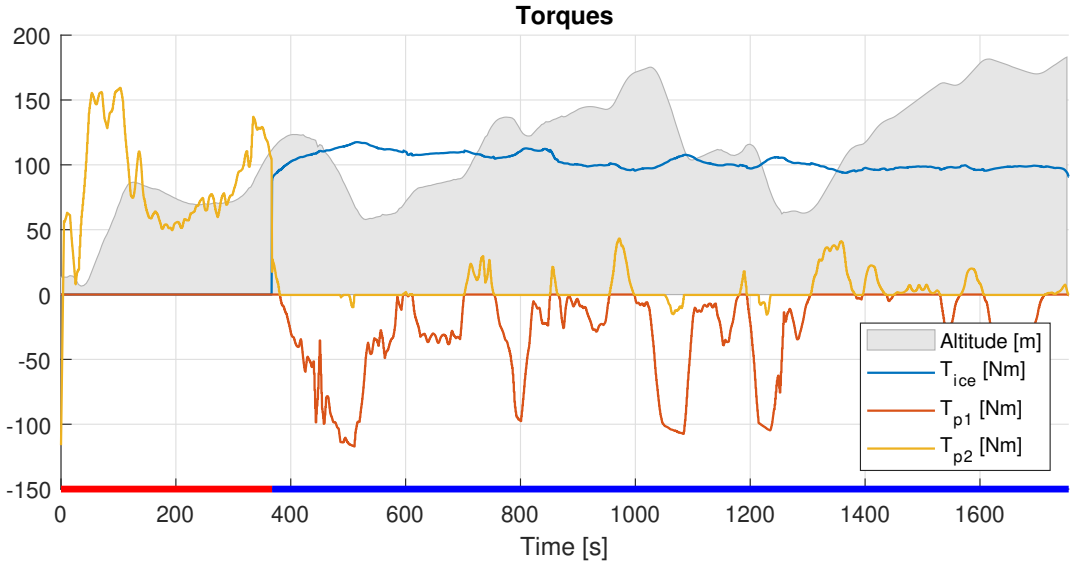


Figure 8.2: Delivered torque of the three power sources along with the altitude profile. Red = EV, Blue = Parallel

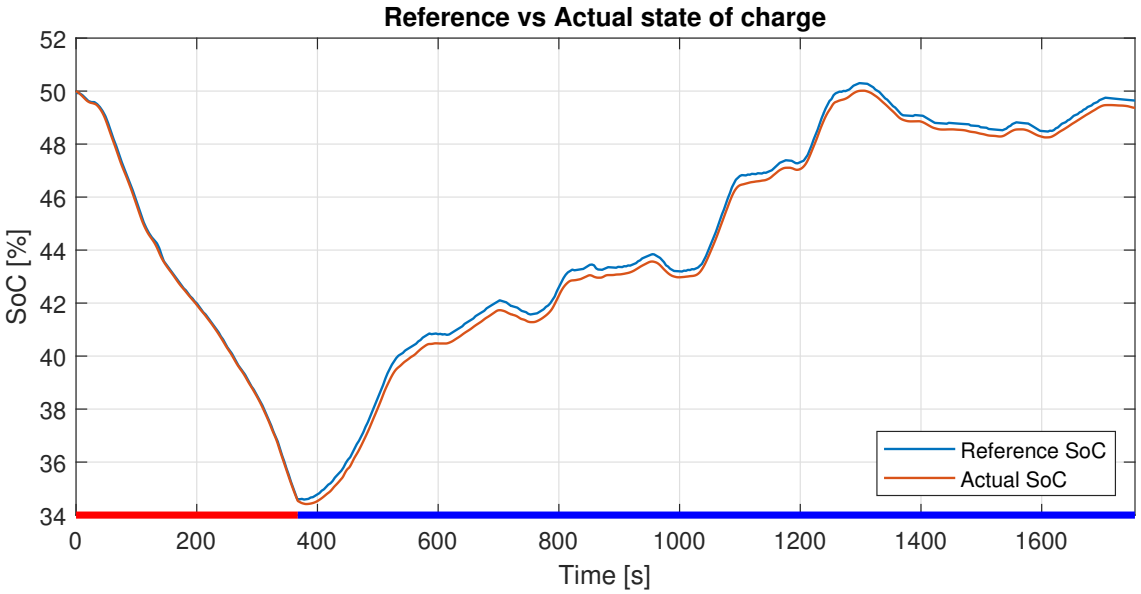


Figure 8.3: Actual and reference SoC trajectories. Red = EV, Blue = Parallel

As can be seen in Figure 8.2, the controller starts by utilizing the EV mode since the SoC reference in Figure 8.3 suggests to drain the battery. At approximately $t = 350\,\mathrm{s}$ the SoC reference changes direction and starts to increase. In order to follow the SoC reference and the relatively high velocity reference in combination with the uphill, the controller selects the Parallel driving mode by delivering the

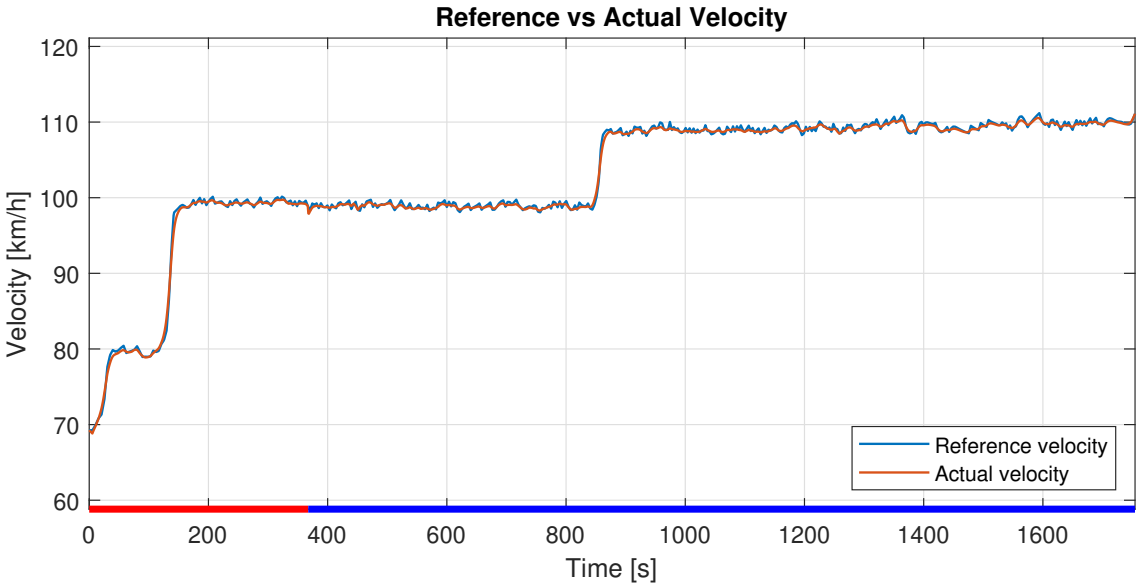


Figure 8.4: Actual and reference velocity trajectories. Red = EV, Blue = Parallel

required torque with the ICE. P1 applies a negative torque to regenerate energy to charge the battery and hence follow the SoC reference. The remaining torque provides the necessary traction force to follow the velocity reference. During parallel mode, some segments can be seen where the P2 motor is involved. In those cases, the P2 motor assists the ICE by delivering the additional torque needed for climbing the hills. Additionally, during steeper downhills, the P2 assists the P1 motor in regenerating when its power limit is being reached.

By regarding Figure 8.3 it can be noticed that the actual SoC profile is below the reference SoC for the majority of the route. This behavior is due to the weighting between the different cost parameters. The SoC tracking cost $K_{\zeta_{\text{err}}}$ tries to achieve $\zeta = \zeta_{\text{ref}}$ which is something that takes energy to achieve if the SoC is below the reference. However, a cost is also placed on the fuel consumption K_{fuel}, which is the main source of energy. These two costs will inhibit each others goal and thus a compromise gets made where the SoC is slightly below the reference.

The computed fuel consumption of this simulation is 0.7361 l/10km and the travel time is 29m 15s. This can be compared to the C1 predicted values of 0.7720 l/10km and 29m 13s. Thus, C2 actually performed better than predicted by C1, which can partly be attributed to C2 being able to make use of parallel drive which C1 cannot. The other reason why C2 is able to improve the performance is due to its ability to make slight deviations from the reference signals where the magnitude of the deviations are determined by the weighting of the cost function. Since C2 also includes a predictive behavior which is based on a more accurate model of the powertrain compared to C1, it is able to notice enhancements which can be made during control which may require small deviations from the reference signals. Hence, by allowing such behavior, the control performance can achieve further optimality which is reflected in the fuel consumption figures.

From Figure 8.4 one of the benefits with a offline-online coupled solution can be

seen. Since the online controller in C2 is allowed to make minor deviations from the reference velocity, the oscillating behavior seen in the reference signal can be eliminated which results in a smoother actual velocity. It can further be noticed that there is a minor drop in the vehicle velocity for a short period of time when the parallel mode is engaged. This is a result of the DE being connected. When the controller switches from EV to Parallel mode, ω_2 which is rotating at high speed is directly attached to ω_1 which is at standstill. By connecting the axes with the DE, a portion of the kinetic energy in the ω_2 shaft is used to accelerate the ω_1 shaft since they must have synchronous speed in the parallel mode. Hence the vehicle speed will drop for a short period of time. The issue with the drop in velocity when the DE connects has hence been observed but not treated.

In addition to developing a functioning solution to the hybrid powertrain energy management problem, it is also of high value to investigate the performance effects of some key components of the solution. The results of this investigation is presented in the following sections.

8.2.2 The effects of election period length on fuel consumption and driveability

This section investigates how the performance of the control solution in terms of relative fuel consumption and drivability is affected by the length of the election period. The controller setup and results presented in Sections 8.1 and 8.2 is considered as the baseline case by which the following tests will be compared to. The performance metric of relative fuel consumption is defined as:

$$\Delta_{\text{fuel}} = 100 \cdot \frac{V_{\text{fuel}}^{\text{T}} - V_{\text{fuel}}^{\text{BC}}}{V_{\text{fuel}}^{\text{BC}}} \, [\%] \tag{8.3}$$

where $V_{\text{fuel}}^{\text{T}}$ denotes the total fuel consumption of the test case and $V_{\text{fuel}}^{\text{BC}}$ denotes the total fuel consumption of the baseline case.

In addition to the baseline case, five additional C2 controllers were created with identical tuning of the cost function but with election periods of 5, 10, 15, 25 and 30 s respectively. Each test case was simulated on the 50 km test route, the fuel consumption and travel time results can be seen in Table 8.2.

Table 8.2: Compilation of consumption results for the test cases and baseline case.

t^{E} [s]	V_{fuel} [l/10km]	Δ_{fuel} [%]	T_s	Δ_{T_s} [%]
30	0.7410	0.6695	29m 15s	-0.0032
25	0.7419	0.7936	29m 15s	-0.0001
20 (BC)	0.7361	-	29m 15s	-
15	0.7360	-0.0193	29m 15s	0.0022
10	0.7360	-0.0193	29m 15s	0.0025
5	0.7380	0.2551	29m 17s	0.1271

From the table it can be concluded that the election period selected as the baseline

case achieves one of the lowest fuel consumptions over the driven route. Based on the relative fuel consumption figures, it can further be concluded that varying the length of the election period does not significantly affect the fuel consumption. However, the behavior of the controllers in terms of control signals is something which changes notably more than the consumption figures when changing the length of the election periods. This behavior is very similar for election periods of length 10 s and above and is shown in Figure 8.5. However, for shorter election periods, such as the simulated case of 5 s, the torque behavior drastically changes, see Figure 8.6.

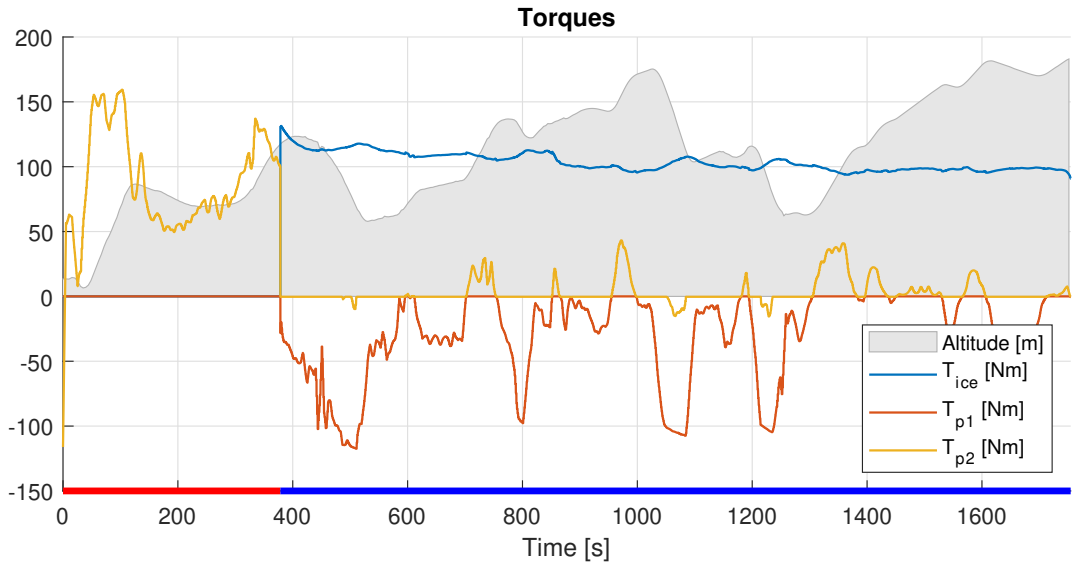


Figure 8.5: Torque inputs for controller with 10 s election period. Red = EV, Green = Serial, Blue = Parallel

Since these drastic changes is not reflected in the fuel consumption figure, other aspects have to be considered in order to give a more fair evaluation of the impact which different lengths of election periods have, such as drivability. The drivability was evaluated in accordance with the metrics stated in Section 6.5 and the results are presented in Table 8.3.

Table 8.3: Values of the drivability metrics for each length of election period. E_{ICE} is the number of engine events. $a^{\max}$ and $j^{\max}$ is the maximum achieved acceleration and jerk magnitude achieved, respectively. $n_{\hat{a}}$ and $n_{\hat{j}}$ is the number of comfortable acceleration and jerk threshold violations done, respectively.

t^{E} s	E_{ICE}	$a^{\max}$ [m/s^2]	$j^{\max}$ [m/s^3]	$n_{\hat{a}}$	$n_{\hat{j}}$
30	2	1.1060	0.4055	0	0
25	4	1.1059	1.6970	0	1
20 (BC)	2	1.1060	2.0753	0	2
15	2	-1.2070	2.5739	0	3
10	2	-1.1476	2.4640	0	2
5	63	-1.2808	2.6719	0	63

As seen in the table, the number of engine events increased drastically for the 5 s

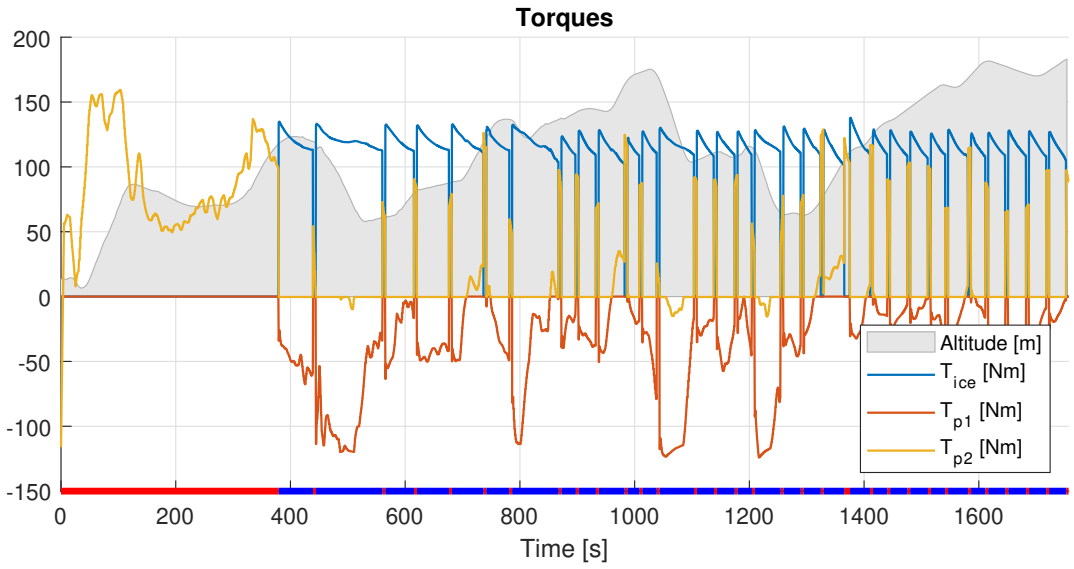


Figure 8.6: Torque inputs for controller with 5 s election period. Red = EV, Green = Serial, Blue = Parallel

election period test. A potential reason for this is that the winning margin criteria is very sensitive to the length of the election period. A longer election period results in a longer election horizon, which results in objective costs of higher value. This means that the relative difference between the costs of the different NMPCs in each election is higher for longer election periods, thus the winning margin must be tuned for each specific election period.

Contrary to the engine events E_{ICE}, the maximum acceleration $a^{\max}$ and jerk $j^{\max}$ does not differ to the same extent for varying lengths of the election period. Furthermore, no controller violates the acceleration comfortability threshold, which is desirable. However, when comparing the number of occurrences of jerks $n_{\hat{j}}$ over the specified threshold, a major difference can once again be seen when comparing the election period of 5 s to the other election periods. It appears that the number of jerk violations correlates with the number of engine events, which proves why the number of engine events is an important metric when considering drivability.

8.2.3 Gearshifting and unexpected stops

An early opinion, stated in the thesis introduction, was that the full control solution benefits more from having the detailed dynamical model in the online component rather than in the offline component. The opinion was that, by doing so, the disturbance robustness would increase. Hence, the developed solution was exposed to a disturbance in the shape of an unexpected stop when simulating on the 50 km test route. The stop was modeled as a timed braking segment where the mechanical brakes supply 600 Nm braking force for a period of 60 s with all motor torques overridden to zero, resulting in a complete stop. When the 60 s have passed, the controller is able to start following the references to recover the deviations.

The controller used in this test was re-tuned in order to produce a gear shift and

hence display that the implemented gear shift functionality works. The baseline controller presented earlier uses a fairly long election period of 20 s and a winning margin of 10 % which is too restrictive for rapid accelerations. A long election period results in few gear shifting opportunities during the acceleration period which means that the chosen gear must be the one that is optimal during the majority of the acceleration period. The optimal gear in that case becomes the highest gear since the reference velocity is a freeway speed, any lower gears would be limited by the power limit in the motor and battery. Hence the controller was given an election period of 2 s for this test. Furthermore, as discussed in Section 8.2.2, the winning margin must be set lower when the election period is shorter in order for the reference tracking to still work, thus it was set to 0 %.

The simulation results can be seen in Figures 8.7 and 8.8. From the figures it can be concluded that the controller is able to recover accurate tracking of the reference signals after the stop. The figures further displays the benefit of utilizing distance based reference signals since they become constant at the point of a complete stop. If time based reference signals were to be used instead, the references would keep changing and effort would need to be put in place in order to make sure the reference stays synchronized with the vehicle position.

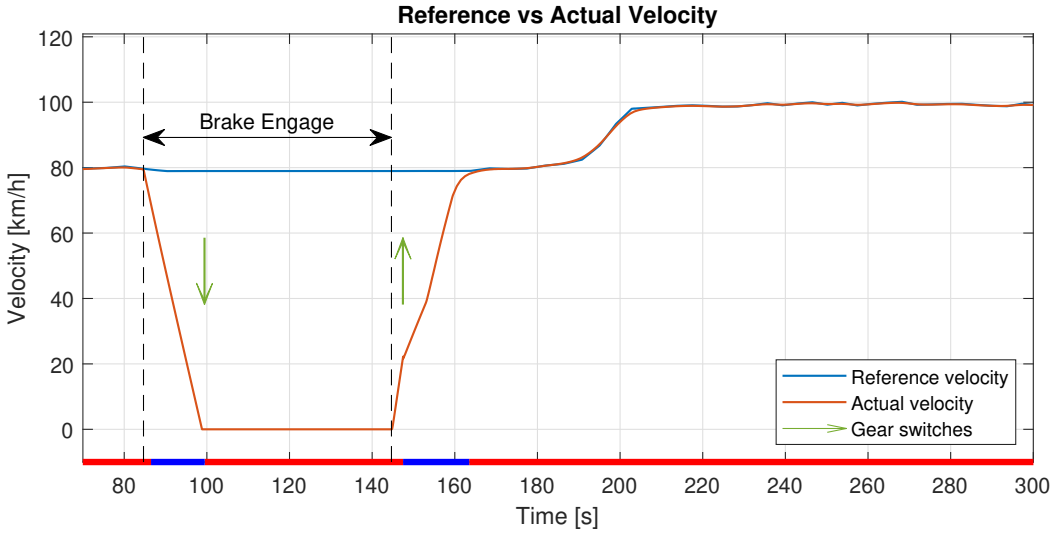


Figure 8.7: Actual and reference velocities during brake test simulation. The vertical arrows represent down and upshift events. Red = EV, Blue = Parallel

8.2.4 The effects of excluding the offline optimization

In order to evaluate the impact of the offline optimization, a simplified controller was designed which does not utilize any offline computed references. Instead the simple online controller follows a constant SoC reference of 50 % and the velocity reference is strictly set to the speed limit of the road. It was noted that upon simulating the simple controllers on the test route, the $K_{\zeta_{\text{err}}}$ cost had to be tuned since the default cost resulted in the controller barely utilizing the battery at all. With a lower $K_{\zeta_{\text{err}}}$ cost, the controller may start to drain the battery below the limit of 20 %, thus an

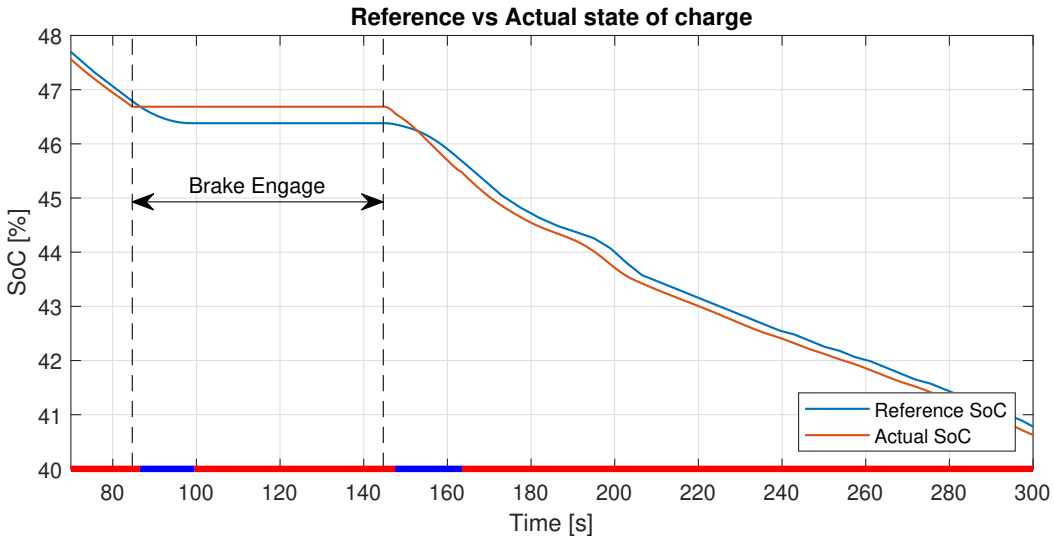


Figure 8.8: SoC from brake test simulation. Red = EV, Blue = Parallel

additional constraint on the lower bound of the ζ value was added to the NMPC controllers for this test. The results from the 50km test route simulations for a range of $K_{\zeta_{\text{err}}}$ is presented in Table 8.4.

Table 8.4: Results from comparing the baseline and the simple non-reference based controller with a set of different $K_{\zeta_{\text{err}}}$.

Controller	$K_{\zeta_{\text{err}}}$	$\zeta_{\text{err}}(T_{\text{s}})$ [%]	$\zeta^{\max} - \zeta^{\min}$ [%]	$v_{\text{err}}^{\max}$ [km/h]	Δ_{fuel} [%]	$\Delta_{T_{\text{s}}}$ [%]	V_{fuel} [l/10km]	T_{s}
Baseline	1e9	-0.63	15.60	2.52	-	-	0.7361	29m 15s
Simple	1e9	-0.26	0.64	6.59	1.8558	-0.6456	0.7498	29m 3s
Simple	1e8	-0.83	1.68	6.53	0.3743	-0.6322	0.7389	29m 4s
Simple	1e7	-3.11	4.01	6.51	0.3846	-0.6426	0.7389	29m 3s
Simple	1e6	-9.79	29.88	8.11	4.4239	-0.5491	0.7687	29m 5s

By regarding Table 8.4, the simple controllers with SoC tuning of 1e9 and 1e8 can be considered to perform equally well as the baseline controller in terms of fuel consumption, travel time and terminal SoC, $\zeta_{\text{err}}(T_{\text{s}})$. However, it should be noted that these simple controllers do not utilize the battery to the same extent as the baseline controller which can be seen by looking at the $\zeta^{\max} - \zeta^{\min}$ metric. The 1e7 and 1e6 case are diverging too far from the desired terminal SoC to be considered as good results. For the controllers with tuning 1e9 and 1e8, the slightly decreased travel time comes with a price of either increased fuel consumption or a greater deviation from the set terminal SoC value. To further evaluate whether or not these simple controllers are equally good as the baseline controller, drivability aspects such as number of engine events should be considered. Engine events is regarded as a key drivability metric since the opinion is that they are highly noticeable by the driver and can be regarded as a distraction when occurring in excess. The number of engine events for the simple controllers with tuning 1e8 and 1e9 are 20 and 14 respectively. The baseline controller however, only contains 1 engine event. This may indicate that a strength of the offline optimization is that it improves drivability in terms of lowering the number of engine events. This would be explained by the expensive engine switching cost in the objective function in C1 while still being able

to achieve the desired terminal SoC value.

To extend the comparison of the controllers, they were simulated on the bowl route described in Section 7.3. The results from these simulations are shown in Table 8.5 as well as Figures 8.9 and 8.10.

Table 8.5: Bowl route simulation results with a set of different $K_{\zeta_{\text{err}}}$.

Controller	$K_{\zeta_{\text{err}}}$	$\zeta_{\text{err}}(T_{\text{s}})$ [%]	$v_{\text{err}}^{\max}$ [km/h]	Δ_{fuel} [%]	$\Delta_{T_{\text{s}}}$ [%]	V_{fuel} [l/10km]	T_{s}
Baseline	1e9	-0.73	0.87	-	-	0.4274	17m 16s
Simple	1e9	-0.39	24.53	9.8310	-8.0030	0.4694	15m 53s
Simple	1e8	-1.66	1.98	0.8238	-1.1016	0.4309	17m 4s
Simple	1e7	-6.15	3.25	-5.5140	-0.1922	0.4038	17m 14s
Simple	1e6	-17.30	0.42	-1.9722	-0.1505	0.4190	17m 14s

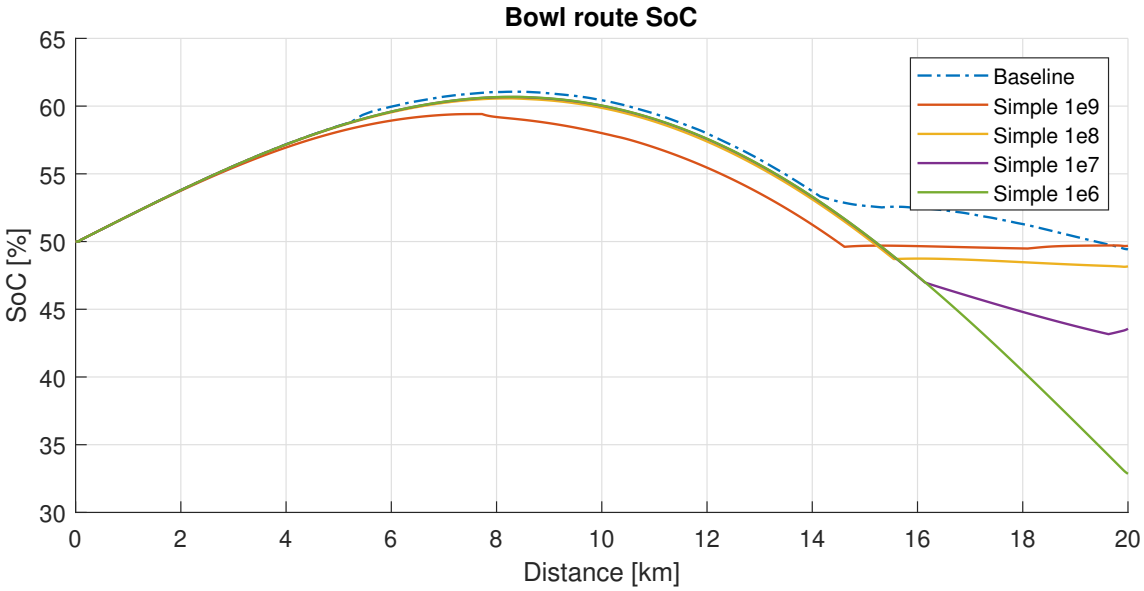


Figure 8.9: Comparing the SoC trajectories between the baseline simulation that follows the C1 reference and the simple controllers that follow a constant reference.

As for the results from the test route, the simple controllers which deviate too much from the terminal SoC $\zeta_{\text{err}}(T_{\text{s}})$ are removed from the evaluation, in this case the simple controllers with tuning 1e7 and 1e6. The simulation results from the bowl route are more dramatic than the results from the test route. For instance, the simple controller with SoC tuning of 1e9 has a significantly lower travel time but with a relative fuel consumption of almost 10 % as can be seen in Table 8.5. However, the simple controller with SoC tuning of 1e8 has a similar behavior as the baseline controller when regarding the SoC and velocity trajectories seen in Figures 8.9 and 8.10.

The considerable decrease in travel time T_{s} and increase in relative fuel consumption Δ_{fuel} produced by the simple controller with SoC tuning of 1e9 is a result of the SoC tracking being too aggressive for this route on a constant reference. The gained energy in the downhill is absorbed using both kinetic and battery energy instead of just battery energy, which results in an increased velocity and thus a shorter travel

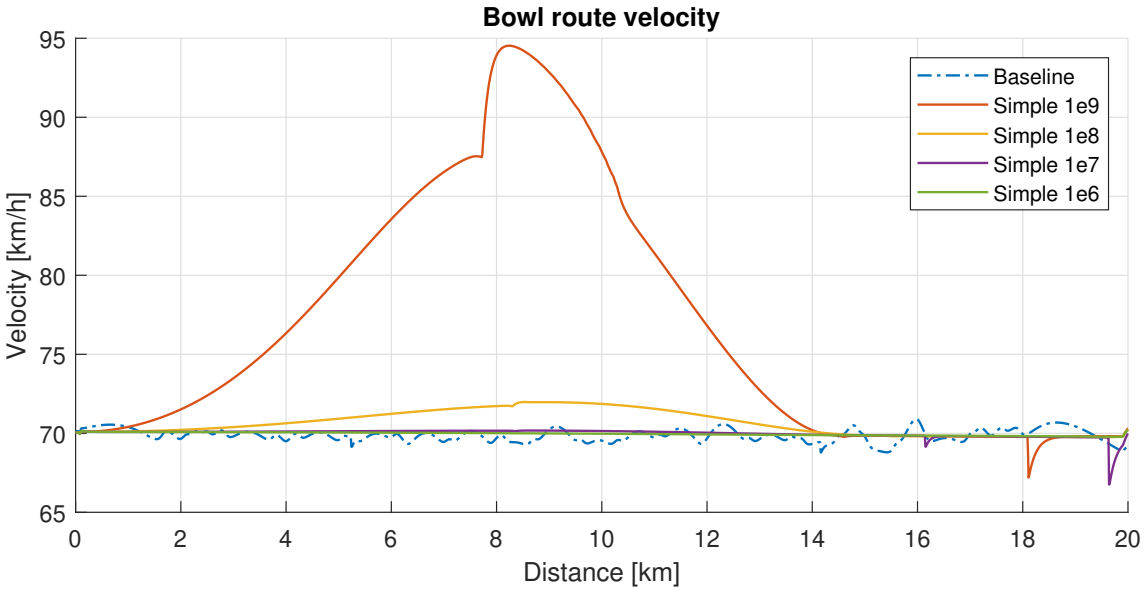


Figure 8.10: Comparing the velocity trajectories between the baseline simulation that follows the C1 reference and the simple controllers that follow a constant reference.

time, which can be seen in the corresponding $v_{\text{err}}^{\max}$ metric in addition to Figure 8.10. Furthermore, it should be noted that gaining kinetic energy by simply rolling down a hill can also be the most energy efficient approach in some situations, such as climbing a hill following a downhill. It is a question of choosing the energy storage method with the minimum overall losses. The decision whether to roll or to use generative braking should be made by regarding the increased loads from drag and friction as an efficiency and then compare to the energy conversion losses that regenerative braking entails. Additionally, such decision is highly dependent on velocity since the aerodynamic load for instance is exponentially increasing with the vehicle velocity. The simple controller with SoC tuning 1e9 is greatly over speeding during the bowl route simulation which results in very large aerodynamic and friction loads. Despite this the controller chooses to roll down the hill instead of using the generative brake which manifests as an increased fuel consumption and a decreased travel time. Since $v_{\text{err}}^{\max} = 24.53\,\text{km/h}$ for this controller and the bowl route uses a speed limit of 70 km/h, the violation from the speed limit is illegal which of course is a downside of simple controller. A mechanical brake would alleviate this issue, but would result in a net energy loss.

In conclusion, despite the baseline controller not having any significant improvement in fuel consumption in this situation, it can still outshine the simple controller due to its ability to plan ahead. Only the baseline controller manages to properly minimize both the $\zeta_{\text{err}}(T_{\text{s}})$ and $v_{\text{err}}^{\max}$ metrics over the bowl route. Furthermore, for the baseline route, the results have shown that the baseline controller achieves equally good travel time, terminal SoC deviation and fuel consumption as the best performing simple controllers but with only 1 engine event which is due to the added cost in the C1 optimization. Hence, other aspects which enhances the online performance can be regarded in this offline optimization as well, such as prioritizing ICE usage

during freeway speeds where the engine noise is less prominent. Furthermore, the ability to plan the SoC allows the offline optimization to fully utilize regenerative braking segments of the route, where a non-planning controller may have already fully charged the battery ahead of the downhill. An additional benefit of being able to plan the route is that the optimizer can generate references which encourage charging the battery ahead of city driving segments, allowing the EV mode to be utilized to a further extent within city borders, which helps reduce emissions and noise pollution.

8.2.5 Emission free zones

A scenario was constructed where the last 5 km of the 50 km test route was flagged as an "emission free zone" where the ICE must be turned off. This was done to highlight one of the earlier mentioned strengths of the offline-online coupled solution, namely that the planning property which C1 offers can be used to ensure that emission free zones are obeyed while respecting the constraints. Through simulations, the baseline controller and one of the best performing simple controller were exposed to this scenario. The most interesting aspect to regard when comparing the two controllers for this scenario is the SoC trajectories. In order to ensure that the emission free zone can be obeyed while also ensuring that the terminal SoC of 50 % is fulfilled, the SoC trajectories must take height in advance. The SoC trajectories for the two controllers can be seen in Figure 8.11.

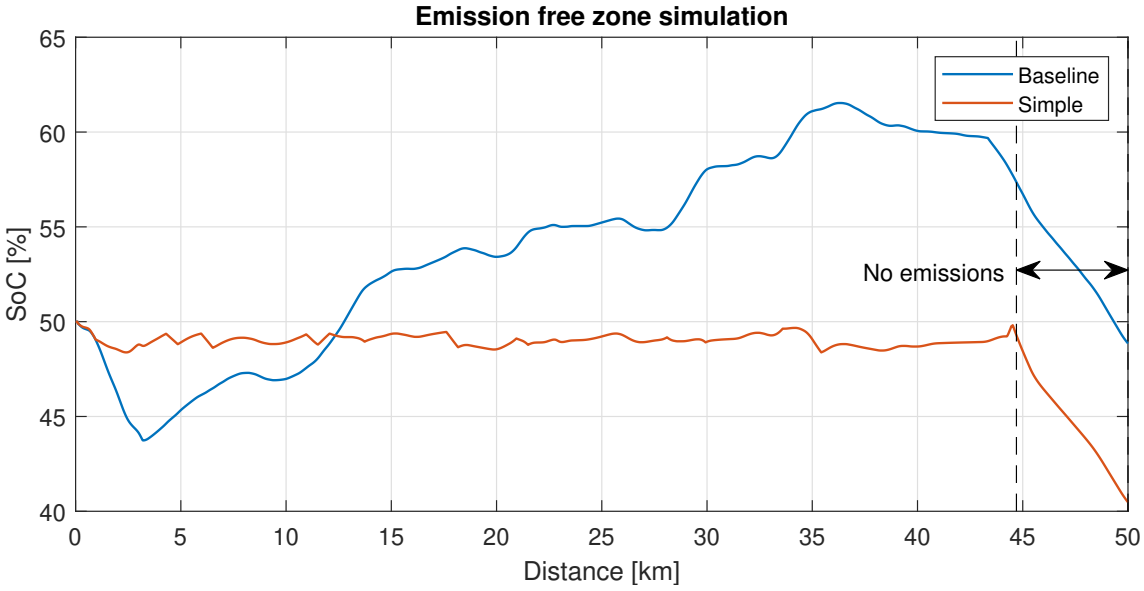


Figure 8.11: SoC trajectories for the baseline controller and the simple controller with SoC tuning of 1e8

The main conclusion which can be drawn from Figure 8.11 is that both controllers are able to obey the emission free zone by draining the battery in EV mode but only the baseline controller manages to achieve a terminal SoC value with almost neglectible difference from the specified value at 50 %. This highlights one of the main benefits with the baseline controller, and hence the coupled solution. By

including the offline optimization, the predictive behavior of the controller is greatly increased. This allows the baseline controller to plan for the emission free zone in time by charging in excess before the emission free zone arrives. The simple controller however, which only relies on a predictive behavior scaled by the fairly short prediction horizon, does not notice the emission free zone in time and is hence not able to increase the SoC trajectory in advance. Since the EV mode is forced within the emission free zone, the battery is drained past the terminal SoC value and thus the simple controller fails to fulfill the terminal SoC constraint.

8.2.6 Responding to driver requested torque

An overtake scenario was simulated on the test route to verify that the technical implementations described in Section 6.3 can treat driver requested torque for such scenario. The driver requested torque was set as $T_d = 300\,\text{Nm}$ between $d = 2 - 3\,\text{km}$ on the route in the simulation. The K_{T_d} cost was set to the rather high value of 1×10^8 in order to make sure that it always has a higher cost than the SoC error cost. By doing so, the tracking of the driver requested torque always has priority over the SoC tracking. This ensures that the driver receives the requested torque in all situations. The SoC error and fuel costs were kept the same as for the previous runs. The simulation result can be seen in Figure 8.12 and 8.13.

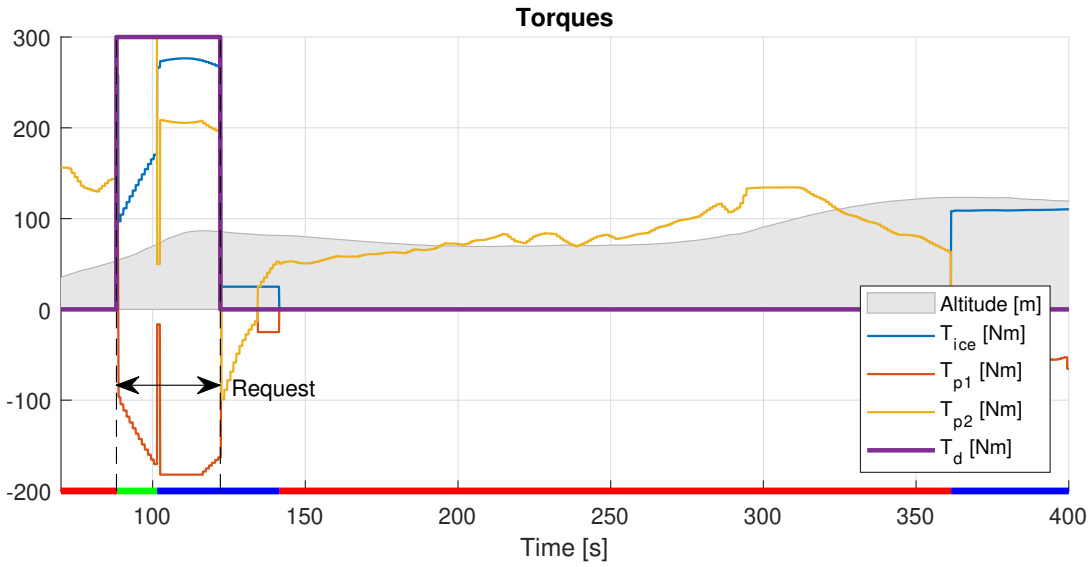


Figure 8.12: Torque inputs of simulation with driver requested torque. Red = EV, Green = Serial, Blue = Parallel

According to the results of the baseline simulation, the EV mode is solely utilized during the first 400 s of the route. However, the simulated overtake scenario includes all three driving modes within the same time segment due to the applied driver requested torque. The request is firstly satisfied by P2 in the serial mode. At roughly 100 s into the simulation, the parallel mode takes over and satisfies the driver requested torque for the remaining period of time with both P2 and the ICE while also charging with P1. The switch from using serial to parallel mode during

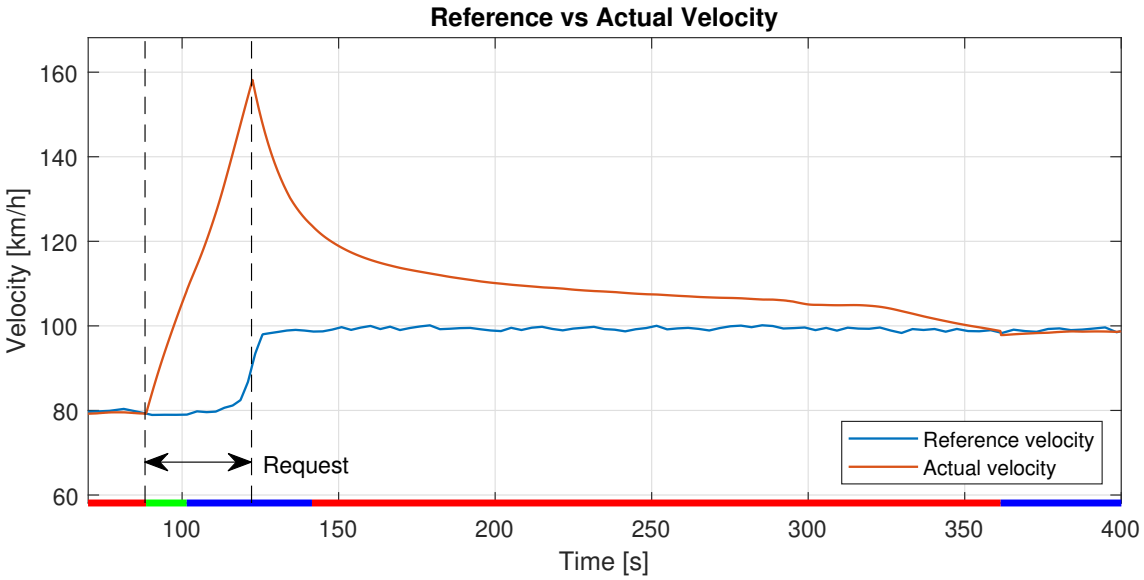


Figure 8.13: Actual and reference velocities with driver requested torque. Red = EV, Green = Serial, Blue = Parallel

the driver torque request can be explained by P1 closing in on its power limit. P2 solely meets the requested torque in serial mode and the power needed is supplied by a combination of the battery power and the generated power from P1. As the power supply from P1 becomes saturated, the remaining part of the P2 power demand has to be supplied by the battery. But since the battery has a power limit, the total power cannot be delivered. Hence, a switch from the serial to the parallel mode is necessary.

Figure 8.13 shows how the velocity deviates from the reference velocity during the driver interaction. Note that the velocity does not rapidly decrease to the reference value when the driver requested torque returns to zero. Instead, the velocity is above the reference velocity for a fairly long period of time. The initial velocity reduction comes from regenerative braking which charges the battery. The regenerative braking is however limited since the controller wants the SoC to track its reference and not be neither above nor below the reference. The cost of the SoC error is now higher than the cost of velocity error, and since no mechanical brake is involved in the model according to the limitations of this project, the vehicle must operate at a velocity higher than the reference.

9

Conclusion

This chapter draws conclusions based on the presented and discussed results in the previous chapter. The conclusions are drawn by answering the formulated research question and concludes the findings in relation to the stated project aim.

According to the aim of this thesis, an online NMPC controller was developed that manages the energy usage and velocity of the hybrid vehicle while tracking offline optimized SoC and velocity references generated by DP. In further accordance to the stated aim, the project has evaluated whether or not the offline optimization contributes to the performance of the solution when compared to solely using the online controller by considering fuel consumption and drivability metrics.

Simulation results show that the coupled solution does not produce better figures on fuel consumption than a standalone 'simple' online controller, however some improvements regarding drivability can be seen. Furthermore, simulations with no emission zones show that the coupled solution is capable of fulfilling such zones while also satisfying the control objective. The 'simple' online controller however showed the capability of complying with such zones but failed to meet the terminal SoC constraint due to its poor planning capabilities. Despite the lack of improvement in fuel consumption, we believe that the coupled solution should not be rejected as a promising solution. The simple offline optimization implemented in this thesis might just be too simple in order to contribute to energy optimization and may benefit a lot from using a more detailed dynamical model of the powertrain. It is further concluded that the coupled solution is fully functional where the online component is able to act on the offline optimized references with the additional functionality of acting on driver requested torque.

In Chapter 1 the main research question was formulated along with three research sub questions. The idea is that by assembling the answers to the sub questions, the main research question can be answered.

How can NMPC be implemented to fit the purposes of hybrid powertrain optimal control?

NMPC can successfully be implemented by using a switched system methodology. A hybrid powertrain involves discrete elements in the dynamics which brings discontinuous behavior. By applying the switched system approach, the discrete elements can be isolated and treated by switching logic, leaving only continuous dynamics within the models accessed by the NMPCs. With only continuous dynamics and the established state equations, objective function and constraints, NMPC proves to be viable for controlling the system.

How can the control solution be adapted to regard driver needs in terms of acting on driver requested torque?

By monitoring for pedal changes, the online controller can switch from tracking velocity to instead track the desired torque from the driver. Furthermore, drivability is considered in multiple ways, as described in Section 6.5, which also leads to the third research sub question below.

How does the choice of horizon lengths influence the overall behavior of the coupled solution in terms of driveability and fuel consumption?

As seen in the results, a short election period and thus horizon length result in worse drivability due to an increased number of mode switches. However, the results indicate a very weak connection between fuel consumption and length of election period. This would probably change by modeling the dynamic components in more detail, such as the disconnect element DE, gear changes and the ICE. These are situations where mode switching losses can be found.

Both drivability and fuel consumption is heavily reliant upon the tuning of the cost function, as is normal in MPC theory. A cost on $\Delta\omega_2$ was necessary in order to remove erratic behavior. Furthermore, the fuel consumption is heavily dependent on how well the controller should track the references. By allowing the controller to deviate more from the references, the controller can take even more efficient decisions, which can result in a lower fuel consumption. However, by doing so exposes the risk of the vehicle not ending up on the desired terminal SoC or that the travel time is increased since it does not follow the velocity reference strict enough.

Main Research Question *What are the strengths and weaknesses of coupling an offline optimization with an online NMPC for hybrid powertrain control in terms of fuel consumption and driveability, compared to using a standalone online NMPC?*

By compiling the answers of the sub research questions and regarding the presented results, the conclusion can be drawn that the developed solution in its current state performs equally well as the standalone online NMPC in terms of fuel consumption and slightly better in terms of drivability. However, the coupled solution revealed a major strength which a standalone NMPC does not provide. The references in the coupled solution are generated based on knowledge from the full route. This allows for greater adaptability where preferences such as terminal SoC value can be regarded and can be enforced to be reached due to the references. This cannot be guaranteed for a standalone online controller which solely relies on a predictive ability scaled by a restricted prediction horizon due to complexity limitations.

Furthermore, with references based on full route knowledge, the most appropriate driving mode can be accessed by the coupled solution at the right time. For instance, with an offline optimized SoC reference, regenerative braking segments can always be utilized since they have been regarded by C1 already at the start of the route. Hence the references formulated by C1 makes sure that the battery is not fully charged at the time of a regenerative braking scenario. Without the offline optimization, such segments can be missed due to lack of planning. The simulation results with a no emission zone further highlighted the strength which the excellent planning capability of the coupled solution brings compared to the standalone online controller. From the simulation results it was noted that the coupled solution obeyed the emission free zone while fulfilling the control objective, which the standalone online controller failed at. These results show that the references greatly influences the behavior of the online controller in the coupled solution. This opens the possibility to make the coupled solution even more sophisticated which is an additional strength of the developed solution. For instance, the SoC optimization can be adjusted to encourage charging with serial or parallel mode at high speeds when the engine sound is less prominent. With such SoC reference, the EV mode can than be utilized to a greater extent during city driving even if the driving segment allows for emissions.

The results of this thesis does not succeed in highlighting any benefits of including velocity as an optimization variable. The cause of this is considered to be the very simple implementation of C1. The belief is that by including a more detailed dynamical model of the powertrain in C1, more relevant velocity references for the particular powertrain can be produced which would have a positive impact on the fuel economy. This belief is based on previous research within predictive cruise control, where for example E. Pearson [20] noted an improvement in fuel consumption for a route with an offline optimized velocity reference.

It should also be highlighted that the coupled solution performs well despite the great simplifications of the powertrain model it uses. Hence, the coupled solution and the concept of utilizing an offline-online control architecture is deemed promising. We therefore suggest further development and testing on C1 with the belief that it can

greatly outperform a standalone NMPC in its final form.

The simulation results further shows that the coupled solution can recover reference tracking after being exposed to unexpected stops and driver interaction. However if such disturbances becomes too challenging, a re-optimization of the reference might be needed which is a disadvantage of the coupled solution since the offline optimization is a time consuming process.

10

Future work

Based on the presented results, discussions and drawn conclusions, the developed energy manager can be further improved by performing a number of tasks. This chapter lists the tasks in an order which regards the value for effort. Meaning that the most value for effort would be achieved by performing the first task in the list.

- Include a detailed dynamical model of the powertrain in C1.
  - Based on the results, it is difficult to improve on the C1 solution in the C2 controller. Hence, a more complex C1 model and optimization is necessary. Particularly, implementing parallel mode would be highly beneficial in the C1 since the C2 controller utilizes that mode a considerable amount.
- Spend more effort on tuning. There are many different parameters to tune and their relationship is complicated, which makes the tuning a time consuming but important task since it was proven that the tuning has a strong impact on the control behavior.
- Use more advanced battery/SoC model. The current model is most likely too simple to be applicable to a real system.
  - Efficiency of the battery is assumed to be 100 %. However, in reality this efficiency is usually in the range of 70-95 % and is also a function of the power charge/drain power and the SoC. By modeling this behavior, a soft limit on the power output of the battery is put in place, which a more intelligent C1 controller could exploit.
  - Tied together with the above point, the current battery model assumes that the battery temperature is constant and thus the controller can demand a high load for an indefinite time. This is not the case since a higher power demand results in a less efficient chemical process which can quickly cause the temperature to rise in the battery, which can have catastrophic consequences if not adequately cooled. By including a model of the battery temperature, a cost in the NMPC cost function could be put in place to prevent high temperatures.
  - The lifetime of the battery is not taken into account at all with the current model. This is a parameter of high importance, and should be a priority in the optimization. By reducing the depth of discharge, and keeping the

battery temperature cool, the lifetime of the battery can be significantly increased.

- Formulate a sophisticated SoC cost function which differentiates between reaching SoC values above and below the SoC reference.
  - As presented in 8.2.6, a flaw in the current SoC cost function can be seen. The controller fails to utilize the regenerative braking ability since the SoC will diverge from the reference SoC if it does that, instead it maintains the vehicle at a high speed in order to bleed off the excessive energy. Obviously, a more optimal solution would be to utilize the regenerative braking all the way down to the reference velocity and then utilize the stored energy later. In order to achieve this, the SoC cost function can be made more complex to allow for less strict reference tracking above the SoC reference compared to below.
- Add a transport delay for the plant input signals in the simulation in order to simulate the computational time of the Controller. This was not investigated in this project since the online computational time was not of focus. However, the delay could potentially have significant impact on the stability and drivability of the solution.
- Implement functionality to re-optimize when unexpected stops makes recovery of reference tracking impossible.
  - Due to the inclusion of a detailed dynamical model of the powertrain in the online component of the energy manager, the presented results show that the solution recovers well from unexpected stops. However, if the length of unexpected stop becomes too large in relation to the length of the route, cases may appear where recovery is impossible. In such situation, it is necessary to allow for C1 to re-optimize the remaining segment of the route as an attempt to enhance the behavior of the controller.
- Implement the ability for C1 to detect segments of city driving in the route, in which it can prioritize EV mode in order to be more prepared for the start-stop behavior in city driving, as well as reduce emissions where it matters the most.
- Implement the ability for C1 to generate an additional reference which states a suggested election period length for driving route.
- Mitigate the issue with the SoC often being slightly below the reference. This is undesirable since generally the SoC should be as high as possible for the solution to be robust to disturbances and unexpected events.

11

References

[1] R. Muncrief, "NO X emissions from heavy-duty and light-duty diesel vehicles in the EU: Comparison of real-world performance and current type-approval requirements Prepared by Rachel Muncrief", Tech. Rep., 2017. [Online]. Available: https://theicct.org/publications/nox-emissions-heavy-duty-and-light-duty-diesel-vehicles-eu-comparison-real-world.

[2] F. Zuccari, F. Orecchini, A. Santiangeli, T. Suppa, F. Ortenzi, A. Genovese, and G. Pede, "Well to wheel analysis and comparison between conventional, hybrid and electric powertrain in real conditions of use", in *AIP Conference Proceedings*, vol. 2191, American Institute of Physics Inc., Dec. 2019, p. 020 158, ISBN: 9780735419384. DOI: 10.1063/1.5138891. [Online]. Available: http://aip.scitation.org/doi/abs/10.1063/1.5138891.

[3] C.-C. Lin, H. Peng, J. W. Grizzle, and J.-M. Kang, "Power management strategy for a parallel hybrid electric truck", *Control Systems Technology, IEEE Transactions on*, vol. 11, pp. 839–849, 2003. DOI: 10.1109/TCST.2003.815606.

[4] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: bridging the gap via the real-time iteration", *International Journal of Control*, vol. 93, no. 1, pp. 62–80, Jan. 2020, ISSN: 0020-7179. DOI: 10.1080/00207179.2016.1222553. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/00207179.2016.1222553.

[5] A. Chasse and A. Sciarretta, "Supervisory control of hybrid powertrains: An experimental benchmark of offline optimization and online energy management", 2011. DOI: 10.1016/j.conengprac.2011.04.008.

[6] W. Yang, J. Yang, J. Liang, and N. Zhang, "Implementation of velocity optimisation strategy based on preview road information to trade off transport time and fuel consumption for hybrid mining trucks", *IET Intelligent Transport Systems*, vol. 13, no. 1, pp. 194–200, 2019, ISSN: 1751956X. DOI: 10.1049/iet-its.2018.5054.

[7] J. Lundgren, M. Ronnqvist, and P. Varbrand, *Optimization*, 3rd ed. Lund: Studentlitteratu, 2008, ch. 18, pp. 481–494, ISBN: 9789144053141.

[8] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, · . Moritz Diehl, and M. Diehl, "CasADi: a software framework for nonlinear optimization and optimal control", *Mathematical Programming Computation*, vol. 11, pp. 1–36, 2019. DOI: 10.1007/s12532-018-0139-4. [Online]. Available: https://doi.org/10.1007/s12532-018-0139-4.

[9] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming", *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, May 2006, ISSN: 00255610. DOI: 10.1007/s10107-004-0559-y.

[10] E. Hellström, A. Fröberg, and L. Nielsen, "A real-time fuel-optimal cruise controller for heavy trucks using road topography information", *SAE Technical Papers*, vol. 2006, no. 724, 2006, ISSN: 26883627. DOI: 10.4271/2006-01-0008.

[11] J. Wollaeger, S. A. Kumar, S. Onori, D. Filev, U. Ozguner, G. Rizzoni, and S. Di Cairano, "Cloud-computing based velocity profile generation for minimum fuel consumption: A dynamic programming based solution", *Proceedings of the American Control Conference*, pp. 2108–2113, 2012, ISSN: 07431619. DOI: 10.1109/acc.2012.6314931.

[12] N. Khaled and B. Pattel, "Theoretical Foundation of MPC", in *Practical Design and Application of Model Predictive Control*, Elsevier, Jan. 2018, pp. 5–16. DOI: 10.1016/b978-0-12-813918-9.00002-2.

[13] D. E. Seborg, "Model Predictive Control 20.1 OVERVIEW OF MODEL PREDICTIVE CONTROL From online version of book by Seborg et al. (2011) on "Process Dynamics and Control"", Tech. Rep., 2011, p. 416.

[14] H. Wang, Y. Huang, H. He, C. Lv, W. Liu, and A. Khajepour, "Energy Management of Hybrid Electric Vehicles", in *Modeling, Dynamics, and Control of Electrified Vehicles*, Elsevier Inc., Jan. 2018, pp. 159–206, ISBN: 9780128131091. DOI: 10.1016/B978-0-12-812786-5.00005-7.

[15] S. Gros, "Direct Optimal Control Introduction", Gothenburg, Sweden: Department of Electrical Engineering, Chalmers University of Technology, 2014.

[16] M. P. Kelly, "Transcription Methods for Trajectory Optimization A beginners tutorial", Tech. Rep., 2015.

[17] S. Gros, "Direct Optimal Control Lecture 4: Shooting Methods Single-Shooting Multiple-Shooting", Gothenburg, Sweden: Department of Electrical Engineering, Chalmers University of Technology, 2014.

[18] L. Zhang, Y. Zhu, P. Shi, and Q. Lu, *Time-Dependent Switched Discrete-Time Linear Systems: Control and Filtering*. Springer, Cham, 2016, vol. 53, ISBN: 978-3-319-28849-9. DOI: 10.1007/978-3-319-28850-5. [Online]. Available: http://link.springer.com/10.1007/978-3-319-28850-5.

[19] T. Levermore, M. N. Sahinkaya, Y. Zweiri, and B. Neaves, "Real-time velocity optimization to minimize energy use in passenger vehicles", *Energies*, vol. 10, no. 1, pp. 1–18, 2017, ISSN: 19961073. DOI: 10.3390/en10010030.

[20] E. Pearson, "Development of a Predictive Cruise Controller for a passenger car using road slope look- ahead information to optimize set speed with respect to fuel-consumption and travelling time", Master's Thesis, University of Stuttgart, 2017, p. 46.

[21] O. Sundström and L. Guzzella, "A generic dynamic programming Matlab function", in *Proceedings of the IEEE International Conference on Control Applications*, 2009, pp. 1625–1630, ISBN: 9781424446025. DOI: 10.1109/CCA.2009.5281131.

[22] I. Buchmann, *How to Prolong Lithium-based Batteries*. [Online]. Available: https://batteryuniversity.com/learn/article/how%7B%5C_%7Dto%7B%

5C_%7Dprolong%7B%5C_%7Dlithium%7B%5C_%7Dbased%7B%5C_%7Dbatteries (visited on Apr. 1, 2020).

[23] W. Zhou, X. Guo, X. Pei, and J. Yan, "Research on Objective Drivability Evaluation with Multi-Source Information Fusion for Passenger Car", pp. 1–7, 2020. DOI: 10.4271/2020-01-1044.

[24] I. Bae, J. Moon, and J. Seo, "Toward a comfortable driving experience for a self-driving shuttle bus", *Electronics (Switzerland)*, vol. 8, no. 9, Sep. 2019, ISSN: 20799292. DOI: 10.3390/electronics8090943.

DEPARTMENT OF ELECTRICAL ENGINEERING
DIVISION OF SYSTEMS AND CONTROL
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY