Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2018

Efficient Route-based Optimal Energy Management for Hybrid Electric Vehicles

Mattias Andreasson and Simon Berntsson

LINKÖPING
UNIVERSITY

Master of Science Thesis in Electrical Engineering

Efficient Route-based Optimal Energy Management for Hybrid Electric Vehicles

Mattias Andreasson and Simon Berntsson

LiTH-ISY-EX--18/5147--SE

Supervisors: **Mahdi Morsali**
ISY, Linköping University
Martin Sivertsson
Volvo Car Corporation
Christoffer Strömberg
Volvo Car Corporation

Examiner: **Lars Eriksson**
ISY, Linköping University

Division of Vehicular Systems
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Abstract

The requirements on fuel consumption and emissions for passenger cars are getting stricter every year. This has forced the vehicle industry to look for ways to improve the performance of the driveline. With the increasing focus on electrification, a common method is to combine an electrical driveline with a conventional driveline that uses a petrol or diesel engine, thus creating a hybrid electric vehicle. To fully be able to utilise the potential of the driveline in such a vehicle, an efficient energy management strategy is needed. This thesis describes the development of an efficient route-based energy management strategy. Three different optimisation strategies are combined, deterministic dynamic programming, equivalent consumption minimisation strategy and convex optimisation, together with segmentation of the input data. The developed strategy shows a decrease in computational time with up to more than one hundred times compared to a benchmark algorithm. When implemented in Volvo's simulation tool, VSim, substantial fuel savings of up to ten percent is shown compared to a charge-depleting charge-sustain strategy.

Acknowledgments

We would like to thank Volvo Car Corporation for giving us the opportunity to write about such an interesting topic. To Martin Sivertsson and Christoffer Strömberg we would like to express our deepest gratitude for all help, support and guidance during the thesis. We would also like to express our gratitude to Viktor Larsson for all discussions and the test drive.

We would also like to thank our thesis colleagues, Moa Källgren and Andreas Hägglund for all the fun times we have had during the writing of the thesis.

From Linköping University, we would like to thank our Supervisor, Mahdi Morsali for the help with the thesis. We would also like to express our gratitude to our Examiner, Lars Eriksson for giving us a solid knowledge and awakened our interest in the subject.

Finally, we would like to thank all our friends and family. Without your support this thesis would not be what it is.

Gothenburg, May 2018
Mattias Andreasson and Simon Berntsson

Contents

Notation **xi**

1 Introduction **1**
1.1 Background 1
1.2 Problem Formulation 2
1.3 Related Research 3
1.4 Objective of the thesis 5
1.5 Delimitations 6

2 Hybrid Electric Vehicle **7**
2.1 A brief introduction to the Hybrid Electric Vehicle 7
2.1.1 Configurations of HEVs 7
2.2 Studied Configuration 10
2.2.1 Component Models 11
2.2.2 Component constraints 14
2.2.3 Vehicle model 14
2.2.4 Operating modes 16

3 Optimisation **19**
3.1 Dynamic Programming 19
3.2 Equivalent Consumption Minimisation Strategy 22
3.3 Convex Optimisation 24
3.3.1 Solvers 24
3.3.2 The dual problem 25

4 Segmentation **27**
4.1 ADASIS 28
4.2 Data 28

5 Method **29**
5.1 General description 29
5.2 Segmentation 31
5.2.1 Data 31

5.2.2 Algorithm . . . 32
5.3 Data setup . . . 33
5.3.1 Operating mode reduction . . . 33
5.3.2 Acceleration between segments . . . 33
5.4 Dynamic programming and ECMS . . . 34
5.5 Convex optimisation, ECOS implementation . . . 34
5.5.1 The objective function . . . 36
5.5.2 Models . . . 37
5.6 Connecting the algorithms . . . 39
5.6.1 Control law . . . 39
5.6.2 Stop criteria . . . 43
5.7 Controller . . . 43
5.7.1 Control method . . . 44
5.7.2 Deviation from trajectory method . . . 45
5.7.3 Deviation from gradient method . . . 46
5.7.4 Implementation in VSIM . . . 47

6 Results 49
6.1 Segmentation algorithm . . . 50
6.2 Optimisation algorithm . . . 52
6.2.1 Drive cycle segmentation . . . 53
6.2.2 Gear reduction . . . 56
6.2.3 Piecewise linear battery model . . . 57
6.2.4 Mean velocity and standard deviation . . . 59
6.2.5 ERAD optimisation . . . 60
6.2.6 Summary . . . 62
6.3 Controller . . . 63
6.3.1 Tuning of the controllers . . . 64
6.3.2 Validation of the controllers . . . 67
6.4 Simulation of complete algorithm with controller . . . 69
6.4.1 Segmentation compared to recorded speed . . . 70
6.4.2 CDCS compared to the implemented algorithm . . . 70
6.4.3 Mean velocity and standard deviation . . . 74
6.4.4 Recalculating the SOC reference trajectory . . . 76
6.4.5 Gear reduction . . . 77
6.4.6 Acceleration segments . . . 78
6.4.7 ERAD optimisation . . . 79
6.4.8 Piecewise linear battery model . . . 80
6.4.9 Summary . . . 81

7 Discussion 83
7.1 Optimisation algorithm . . . 83
7.2 Simulation of complete algorithm . . . 83
7.3 Data accuracy . . . 84
7.4 Vehicle model . . . 84
7.5 Connecting DDP/ECMS and ECOS . . . 85

7.6 Detailed discussion of results 85
7.6.1 Test set 85
7.6.2 Drive cycle segmentation 86
7.6.3 Controller 86
7.6.4 Mean velocity and standard deviation 86
7.6.5 Acceleration segments 87
7.6.6 ERAD optimisation 87
7.6.7 Gear reduction 88
7.6.8 Piecewise linear battery model 88
7.7 Reduction of emissions 88

8 Conclusions and future work 91
8.1 Conclusions 91
8.2 Future work 93

A Drive Cycles 97
A.1 SHC 97
A.2 GAC 98
A.3 PFC 99
A.4 WLTP 100

Bibliography 101

Notation

Vehicle and component model notations

Variable	Meaning
$m_{vehicle}$	Vehicle mass
$v_{vehicle}$	Vehicle speed
$a_{vehicle}$	Vehicle acceleration
J	Inertia
θ	Road inclination
$\omega_{component}$	Angular velocity for the component
$P_{component,mech}$	Mechanical power in the component
$P_{component,loss}$	Power loss in the component
η_{GB}	Gearbox efficiency
I	Current
R_i	Internal resistance in the battery
U_{oc}	Open circuit voltage in the battery
U_{R_1}	Voltage over the internal resistance
SOC	State of Charge for the battery
Δt	Time step
Q_0	Battery nominal capacity
SOC_{start}	The start value of the State of Charge
SOC_{end}	The end value of the State of Charge
Q_{fuel}	The fuel energy

Optimisation notations

Variable	Meaning
λ	The equivalence factor (dual variable)
λ_{RMS}	The root mean square between the input and output λ
q	The weight factor for the update law
n_{seg}	Number of segments
k	A set of equations
s	The weight in the objective function
t_i	The segment i
t_N	The last segment
T_{N+1}	The time when the last segment ends

Segmentation notations

Variable	Meaning
σ	The standard deviation
X_i	The value in each sample point
$\overline{X}$	The mean over all samples
n	The number of samples

Controller notation

Variable	Meaning
$SOCdiff\%$	The difference in the last time step between the actual SOC and the reference SOC trajectory.

Abbreviations

Abbreviation	Meaning
ADAS	Advanced Driver Assistant Systems
ADASIS	Advanced Driver Assistant Systems Interface Specifications
CDCS	Charge-Depleting/Charge-Sustaining
DDP	Deterministic Dynamic Programming
DP	Dynamic Programming
ECMS	Equivalent Consumption Minimisation Strategy
ECU	Engine Control Unit
EMS	Energy Management System
ERAD	Electric Rear Axle Drive
HEV	Hybrid Electric Vehicle
ICE	Internal Combustion Engine
ISG	Integrated Starter Generator
MPC	Model Predictive Controller
PHEV	Plug-in Hybrid Electric Vehicle
PMP	Ponytragin's minimum principle
SOC	State of Charge
SOCP	Second Order Cone Program
SPA	Scalable Product Architecture
VCC	Volvo Car Corporation
VSIM	Volvo Simulation Tool

1 Introduction

1.1 Background

The requirements on fuel consumption and emissions for passenger cars are getting stricter every year. This has forced the vehicle industry to look for ways to improve the performance of the driveline. With the increasing focus on electrification, a common method is to combine an electrical driveline with a conventional driveline that uses a petrol or diesel engine, thus creating a hybrid vehicle. This gives the opportunity to power and propel the vehicle with either electricity from a battery, liquid fuel or a combination of these.

The benefit from having a combustion engine with long range, quick and well-developed infrastructure for refuelling is combined with the energy efficient electric motor. Especially when driving in the city at lower speeds and with many of decelerations that can be used for charging the battery, the electric driveline is highly effective compared to the combustion engine driveline. One big drawback of the electric driveline is however the battery. The battery size has a strong impact on the weight of the vehicle, it consumes a lot of space and it is expensive, consequently the capacity is limited.

With a limited battery, a problem to solve is how to optimally propel the vehicle. To solve this problem, computationally heavy methods such as Dynamic Programming (DP) have been utilised in research. These methods use data about the road profile that are sampled from test driving or generic drive cycles. Another method of accessing road profile data could be from the navigational system, as when driving today, the destination is often set in the navigation system or can be predicted from the driver's behaviour, giving access to road profile data in advance.

Methods for energy management in vehicles today, in general, use minor knowledge about the full route ahead. An improved method can reduce the fuel consumption by using more information about the full route ahead, which is beneficial for both users and the environment. Since vehicles have limited capacity of processing data, an efficient optimisation strategy that is still able to converge close to a global optimum would make an excellent compromise. Due to this, Volvo Car Corporation (VCC) has outlined an interest to further examine the area of optimal control in the Energy Management System (EMS) for Hybrid Electric Vehicles (HEVs). With VCCs current strategy on using electric motors in all future cars and an annual sale of more than half a million vehicles, an improved strategy has the potential for extensive fuel savings.

1.2 Problem Formulation

To be able to fully utilise the potential of a HEV, the EMS has to be designed with many factors in mind. As the battery and the electrical energy often is a limiting factor, the EMS needs to deploy the electrical energy during the sections of the route where using it will decrease the fuel consumption the most. There are however some limiting factors which makes it impossible to deploy the most exact algorithms to decide the optimal solution for the route. The EMS needs to be implemented online in the vehicle where the Engine Control Unit (ECU) often have a very limited amount of computing power and memory. The cloud can be used to increase the computational power, however with a fleet of several million vehicle, the capacity in the cloud is also limited.

A simple and common strategy used in EMS for HEVs is Charge-Depleting/Charge-Sustaining (CDCS), where the vehicle is operated almost entirely with electrical power until the battery level reaches a lower threshold. For a route shorter than the electric range, this strategy is optimal in terms of saving fuel. For a drive longer than the electric range, CDCS is most likely not an optimal solution, as it uses all of the electrical energy directly and not when the benefit is greatest. An illustration of the CDCS method can be seen in Figure 1.1.

This thesis aims to design and evaluate an efficient optimisation-based EMS for HEVs. Implementing a new EMS might lead to reduced costs and a better use of electrical energy. However, the new EMS is subject to the same constraints as the current one, meaning that a new algorithm should be very efficient in terms of computing power and memory usage. As the navigation systems in the cars can provide data about the inclination, speed limits and length of the planned route, that data can be used in the EMS to be able to optimise the deployment of electrical and liquid energy by optimising the torque split between the electrical machines and the combustion engine for the planned route, resulting in an optimal State of Charge (SOC) trajectory to follow.

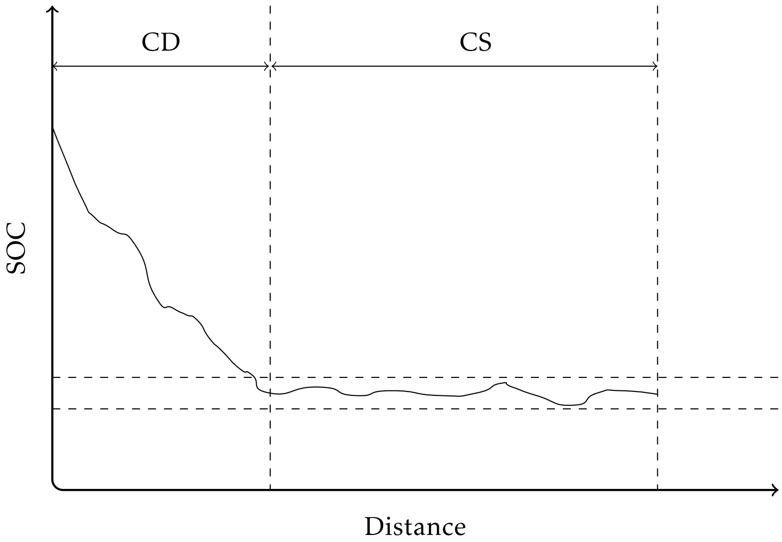


Figure 1.1: An illustration of the CDCS *method. The first region is the CD region, where the battery charge is depleted, using only the electrical machines to propel the vehicle for as long time as possible. In the second region, the CS region, the propulsion is done mainly using the* ICE, *with the electrical machines still recouping battery with regenerative braking and helping the* ICE *when power to the battery has been recovered, to stay in the sustain area as shown with the two horizontal dotted lines.*

1.3 Related Research

There are many different control strategies used to control a HEV. Extensive research has been conducted by a number of authors in this area. Different approaches can be summed up into two sub-categories, rule-based and optimisation-based[1, 2], as seen in Figure 1.2.

Rule-based strategies is defined by sets of rules, that if fulfilled, or not fulfilled will decide in what mode the HEV will be working. The rules are most often set by a method of combined heuristics, intuition, human expertise and powertrain characteristics[1–3], specified for an optimal solution for individual components and thus not the optimum for the entire system. This is however a computational efficient method that can provide a quite good strategy in cases where the rules are set for the correct scenario[2].

The optimisation-based strategies rely on minimising a cost function and can be divided further into two sub-categories, global optimum strategies and real time control strategies[1–3]. Global optimum strategies are formulated to find the global optimum solution for a whole drive cycle. For this purpose, data about the drive cycle needs to be provided in advance[1]. Characterising for such al-

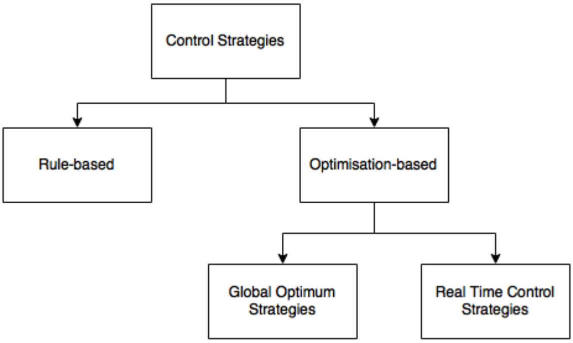


Figure 1.2: *Subcategories of control strategies*

gorithms are the comprehensive amount of computational power required, thus not suitable for usage in vehicles.

DP is a very common method to do offline optimisation. The method is computationally heavy, suffering from the "curse of dimensionality" [1], meaning the computational complexity of the algorithm will increase exponentially with the number of variables. It does however always find a global optimum for the given discretisation. If the discretisation is small, the solution is accurate. A larger discretisation results in a larger error, although with the benefit of reducing the computational time. The method is for those reasons well suited to use as a benchmark algorithm for comparing to other methods[4] as done in [5].

As the issues with Dynamic Programming needs to be accounted for, the case when the disturbances are known in advance have been given a specific name and are referred to as Deterministic Dynamic Programming (DDP)[4]. DDP is well suited for situations where the route, together with speed limits and road inclinations are known in advance, as those are what is referred to as disturbances.

Another of the most researched global optimisation methods for energy management in HEVs is Ponytryagin's minimum principle (PMP)[1]. The method is rather computationally heavy, however it can be combined with piecewise linear approximations of the vehicle model, resulting in a faster algorithm [6].

Convex optimisation is also an interesting for developing EMS. This method does not have the possibility to solve discrete problems such as engine on/off or gear choice, it can however be used to solve the torque split. Convex optimisation is therefore often coupled with either DP [7] or PMP [8, 9]. There is also a possibility to couple it with rule-based strategies as done in[10]. The two optimisation-based methods achieve a good balance between computational time and accuracy in the respective articles[7–9].

The real time control strategies do not need as much computational power and are implementable in a vehicle for real time usage. Of these optimisation strategies, one of the most popular is the Equivalent Consumption Minimisation Strategy (ECMS) [1]. The ECMS strategy is similar to PMP, however where PMP solves the global problem, ECMS can be implemented to solve parts of the optimisation problem in real time[11].

The coupling of the previous stated methods with some sort of data segmentation is an interesting area. The Eco-Discharge strategy presented in[12] shows that the use of segmentation could be a powerful tool. The Eco-Discharge strategy relies on the road profile being accessed by the Advanced Driver-Assistant Systems Interface Specifications (ADASIS)-protocol, the protocol is well suited for usage in HEVs[13]. ADASIS can provide information about the road ahead directly to the CAN-BUS in the car for a simple integration.

As most global optimum strategies are not directly applicable for online implementations, a combination with a real time control strategy is often used. It defines an instantaneous cost function which is possible to solve with a real time solver[1], thus reducing the computing effort needed in the ECU. As the real time control strategies are implementable online, they are important to consider. The most researched of the real time control strategies are the ECMS strategy and the Model Predictive Controller (MPC) strategy[1]. Optimisation strategies will be further discussed in Chapter 3.

1.4 Objective of the thesis

This thesis aims to develop and evaluate an efficient optimisation-based energy management strategy for a HEV using Volvo's Scalable Product Architecture platform (SPA). The SPA platform with a hybrid powertrain has one petrol combustion engine and two electrical machines.

The EMS should use convex optimisation and dynamical programming together with ECMS and segmentation. The thesis is also going to evaluate this EMS against other solutions on the same platform, both in terms of fuel consumption and computational time.

- Can an efficient EMS that generates a correct SOC reference from limited segmented input data be constructed by introducing convex optimisation instead of using a root finding method in DDP/ECMS.
- How much computational time can be saved by segmenting the input data to the optimisation algorithm.
- How well does the algorithm with segments as input perform compared to an optimal solution for the problem without segmented data.
- What variables are needed in the segments to get a correct reference.

The EMS developed during the thesis will furthermore be implemented in VCCs simulation tool, Volvo simulation tool (VSIM). This is made to validate against more advanced models and for benchmarking against CDCS to see if an improvement is possible.

1.5 Delimitations

The mathematical model of the vehicle studied is simplified to exclude any dynamics. Since the optimisation is conducted on a segmented drive cycle, the length of each segment is of importance. Using rather long segments will result in the vehicle is operated at constant conditions for a long time and the dynamics between the segments have minor impact.

The code for some of the optimisation algorithms is not created from scratch, due to this thesis being based on similar algorithms developed in a thesis at VCC [14].

To solve a convex problem, a variety of different solvers exist. Due to the limited time only a few solvers are studied.

The developed EMS is tested only in model-based simulations. Implementation and testing in a real vehicle is not possible given the short period of time and lack of computational power in today's vehicles.

No account is taken to factors such as driveability.

No account is taken to the emissions. The optimisation will only be done with regards to the fuel consumption. Emissions is however related the the fuel consumption, which means that a decrease in fuel consumption will most likely lead to a decrease in emissions.

A limitation in what to present in the report has to be drawn. The possibility for more testing and validation is extensive.

2

Hybrid Electric Vehicle

2.1 A brief introduction to the Hybrid Electric Vehicle

The HEV have become quite common in recent years. This section gives a short explanation about different types of HEVs and how they work.

2.1.1 Configurations of HEVs

There are two different ways to build the HEV in terms of battery charging. The first way is often simply called a HEV. This one does not have external charging possibilities, instead charging takes place through the engine or with regenerative braking. The other way is called a plug-in hybrid electric vehicle (PHEV). This vehicle comes with the same possibilities as a HEV, however also with the added functionality of external charging of the battery by connecting to a charger[15].

Further, there are some different configurations to couple the electrical machine(s) and the combustion engine in the powertrain. There are mainly two different configurations, a parallel powertrain and a series powertrain. These can also be combined, creating a series/parallel powertrain[16]. A presentation of how the different configurations are designed follows.

Parallel powertrain

The parallel powertrain usually consists of an electrical machine and a combustion engine working in parallel, both connected to the same torque split coupling which is then connected to the gearbox[17]. The parallel powertrain has one degree of freedom, meaning that the electric machine and the engine is running at equal angular velocity set by the gearbox and only the torque split is decided

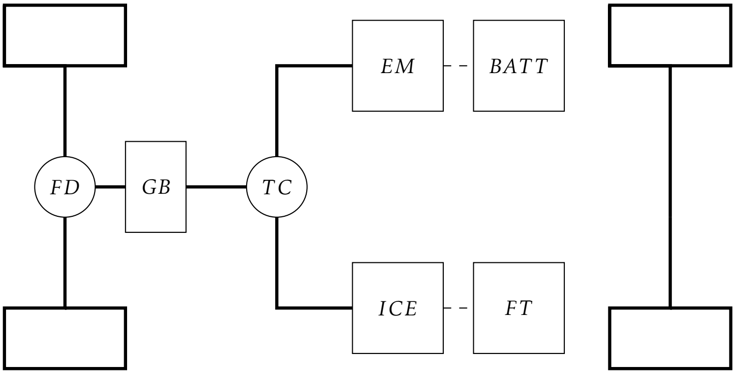


Figure 2.1: *A schematic illustration of a parallel powertrain. The powertrain consists of a final drive (FD), a gearbox (GB), a torque coupling (TC), an electrical machine (EM), a battery (BATT), an internal combustion engine (ICE) and a fuel tank (FT).*

in the EMS [18]. A schematic illustration depicting a parallel powertrain can be found in Figure 2.1.

Through the Road
A more uncommon parallel configuration is the "Through the Road" configuration, which also falls under the parallel powertrain. This configuration consists of two mechanically separated powertrains, where the combustion engine is connected to one of the vehicles axles, while the electric machine is connected to the other axle. This removes the need for a complicated and expensive torque coupling[19]. A schematic illustration depicting a Through the Road powertrain can be found in Figure 2.2.

Series powertrain

The series powertrain consists of a combustion engine, coupled with a generator charging the battery. The combustion engine can when coupled this way be configured to run at a more optimal efficiency, charging the battery through the generator or producing electrical energy directly to the electric machine. The electric motor is connected to the drive shaft. In this configuration, the combustion engine has no direct coupling to the wheels[17]. This gives the series powertrain one degree of freedom from the power link and the ICE can be seen as half a degree of freedom since the requested power can be generated with any combination of torque and angular velocity[18]. A schematic illustration depicting a series powertrain can be found in Figure 2.3.

Series/parallel powertrain

A series/parallel powertrain is a combination of the series powertrain and the parallel powertrain, meaning that it in some way is possible to use the combustion engine for propulsion as in the parallel configuration, while it is still possible to

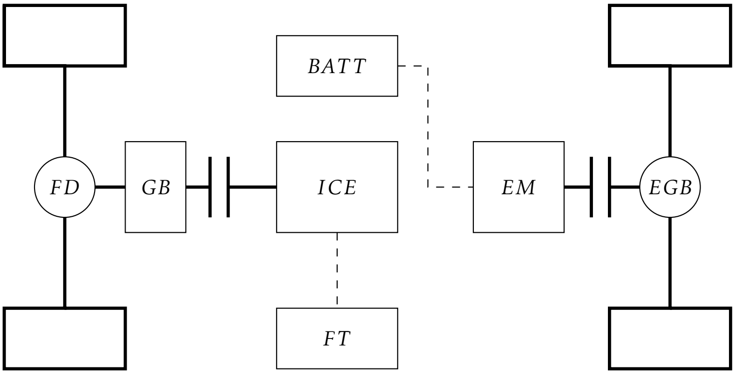


__Figure 2.2:__ A schematic illustration of a Trough the Road powertrain. The powertrain consists of a final drive (FD), a gearbox (GB), an internal combustion engine (ICE), a fuel tank (FT), an electrical machine (EM), an electrical gear box (EGB) and a battery (BATT). There are also two clutches, one between the ICE and the GB, and one between the EM and the EGB.

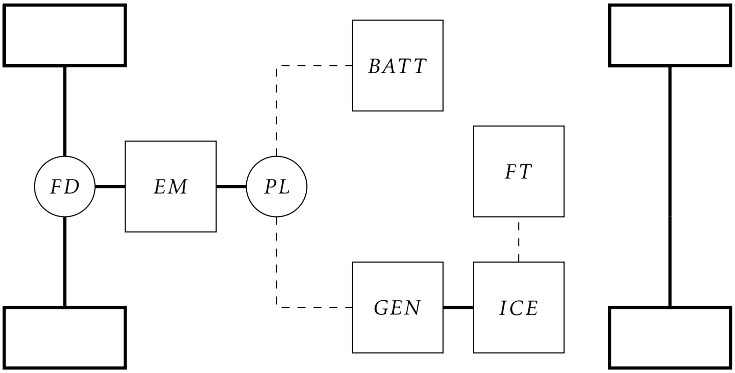


__Figure 2.3:__ A schematic illustration of a series powertrain. The powertrain consists of a final drive (FD), an electrical machine (EM), a power link (PL), a battery (BATT), a generator (GEN), an internal combustion engine (ICE) and a fuel tank (FT).

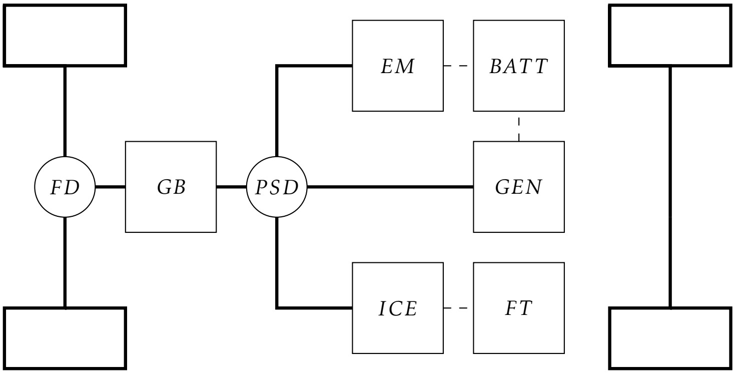


Figure 2.4: *A schematic illustration of a series/parallel powertrain. The powertrain consists of a final drive (FD), a gearbox (GB), a power split device (PSD), an electrical machine (EM), a battery (BATT), a generator (GEN), an internal combustion engine (ICE) and a fuel tank (FT).*

use the ICE for producing electrical energy through a generator, as in the series configuration. There are different ways to produce this coupling[16, 17]. The series/parallel powertrain has two degrees of freedom, due to to separate powertrains that can be controlled individually [18]. A schematic illustration depicting a series/parallel powertrain can be found in Figure 2.4.

2.2 Studied Configuration

The studied configuration in this thesis is the SPA platform from VCC. It is a PHEV with a combined parallel and through the road configuration. The parallel configuration is connected to the front axle, where an internal combustion engine (ICE) is coupled with an Integrated Starter Generator (ISG). The ISG replaces the standard starter motor and can act as both motor and generator. These are directly connected to the motor shaft, meaning that the ISG and the ICE must be turning at the same time, with equal rotational speed. This gives the possibility to charge the battery through the ISG at the same time as running the ICE, however also the limitation that there is no possible way to use the ISG for full electric propulsion.

For full electric propulsion, the Electric Rear Axle Drive (ERAD), which is an electrical machine connected to the rear axle, is used. The machine can give full electric propulsion to the vehicle through rear wheel drive. Between the ERAD and the rear axle is a single ratio electrical gearbox, which can be disconnected from the ERAD with a clutch when not in use, for the purpose of minimising losses.

With this configuration, the possibility to use a four-wheel drive is given using the ICE/ISG and the ERAD at the same time. A schematic illustration of the studied SPA platform can be found in Figure 2.5.

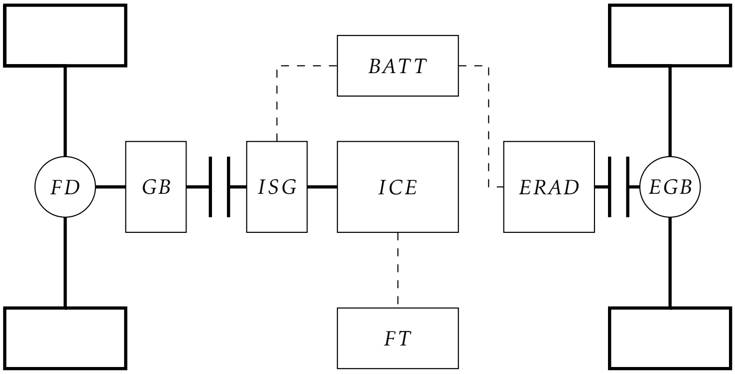


Figure 2.5: *A schematic illustration of the configuration of the studied vehicle. The powertrain consists of a final drive (FD), a gearbox (GB), an Integrated Starter Generator (ISG), an internal combustion engine (ICE), a fuel tank (FT), a battery (BATT), an Electric Rear Axle Drive (ERAD) and an Electrical Gearbox (EGB). There are also two clutches in the powertrain, one between the ISG and the GB and one between the ERAD and EGB, making it possible to connect or disconnect any of those sides of the powertrain.*

2.2.1 Component Models

The losses of the combustion engine and of the electrical machines are modelled as piecewise linear equations. These equations are written as Equation 2.1, where k represents the set of linear equations for different loads and a specific angular velocity ω. The linearisation is made by VCC based on measurements.

$$P_{loss}^{k}(\omega) = A^{k}(\omega) \cdot P_{mech} + B^{k}(\omega) \tag{2.1}$$

The number of angular velocities where the lines are defined are quite few to reduce the data required. Interpolation is used to find the correct set of lines given the current angular velocity. The actual loss in the component has to be equal to the maximum value of all linear, see Equation 2.2. Figure 2.6 illustrates a load with blue line and the loss as the blue circle, which is the maximal value of piecewise linear functions along the line. As seen, the value differ slightly from the actual value on the black line. With increasing number of lines, that difference can be reduced. A general description of the function to find the loss is given in Equation 2.3.

$$P_{loss} = \max_{k}(A^{k}(\omega) \cdot P_{mech} + B^{k}(\omega)) \tag{2.2}$$

$$P_{loss} = f(P_{mech}, \omega) \tag{2.3}$$

With this way of modelling, all dynamics are removed from the component mod-

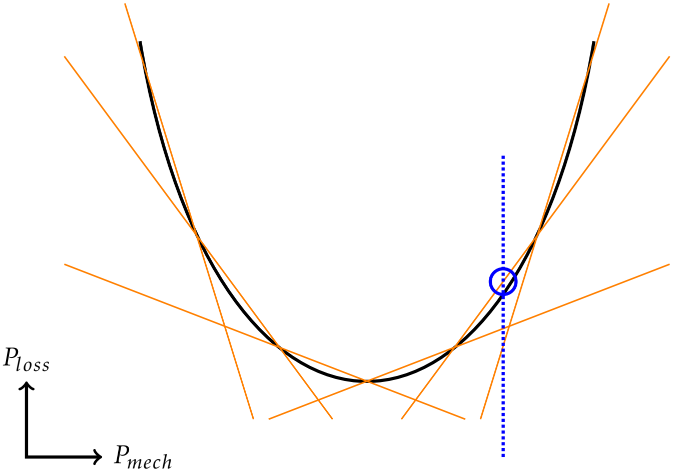


Figure 2.6: Piecewice linearisation illustrated. The black line in the illustration depicts a function, the orange lines depicts the piecewise linearisation of the function with 6 lines and the blue line and circle depicts the piecewise approximation of the function at a certain place.

els. For each time step, the losses are calculated only depending on the current load and angular velocity. Since the models have no transient behaviour, the accuracy might differ from a more complex model. However, to be convinced that the model is convex and possible to solve with limited resources, a choice of a higher complexity is not justified.

Electric machines and combustion engine

The following functions are given for the combustion engine and electrical machines.

$$P_{ICE,loss} = f(P_{ICE,mech}, \omega_{ICE}) \tag{2.4}$$

$$P_{ISG,loss} = f(P_{ISG,mech}, \omega_{ISG}) \tag{2.5}$$

$$P_{ERAD,loss} = f(P_{ERAD,mech}, \omega_{ERAD}) \tag{2.6}$$

Gearbox

The gearbox is modelled with losses proportional to the load and choice of gear, where η_{gb} is the loss coefficient for a specific gear. The load $P_{mech,inputshaft}$ is the sum of the power from the ICE and ISG, since these are connected to the output shaft of the engine.

$$P_{GB,loss} = |\eta_{gb}(gear) \cdot P_{mech,inputshaft}| \tag{2.7}$$

$$P_{mech,inputshaft} = P_{ICE} + P_{ISG} \tag{2.8}$$

When the absolute sign is removed, the losses can be described with a piecewise linear approach as the maximum of the positive and negative value of Equation 2.7.

$$P_{GB,loss} = \max(\pm\eta_{gb}(gear) \cdot P_{mech,inputshaft})) \tag{2.9}$$

Electrical gearbox

For the electrical gearbox, the loss coefficient η_{egb} is modelled dependent on the angular velocity of the rear axis and the direction of energy transfer. A map with a set of velocities where interpolation and if necessary extrapolation is used.

$$P^k_{EGB,loss} = \begin{cases} \eta^k_{egb}(w_{wh}) \cdot P_{mech} & k = 1, P_{mech} \geq 0 \\ \eta^k_{egb}(w_{wh}) \cdot P_{mech} & k = 2, P_{mech} < 0 \end{cases} \tag{2.10}$$

This can also be described as a piecewise linear function.

$$P_{EGB,loss} = \max_k(P^k_{EGB,loss}) \tag{2.11}$$

Battery

The battery is modelled as a voltage source U_{oc} and an internal resistance R_i. Since the complexity of the problem to solve is desired to be low and the models for the rotating components are highly simplified, this model is regarded as a good choice.

Combining the equation for power and Ohms law, the power loss can be solved to depend on the current and resistance, see Equation 2.14. U_{R_i} is the voltage over the resistance and I the current in the circuit.

$$P_{batt,loss} = U_{R_i} \cdot I \tag{2.12}$$

$$U_{R_i} = R_i \cdot I \tag{2.13}$$

$$P_{batt,loss} = R_i \cdot I^2 \tag{2.14}$$

The voltage source can be modelled as a constant or to be changing with the SOC, Equation 2.15.

$$P_{ech} = U_{oc}(SOC) \cdot I \tag{2.15}$$

Solving Equation 2.15 for current and combining with Equation 2.14, yields an expression for the power loss of the battery, Equation 2.16.

$$P_{batt,loss} = \frac{R_i}{U_{oc}(SOC)^2} {P_{ech}}^2 \tag{2.16}$$

As seen, the loss depends on the square of the voltage. To avoid modelling multiple squares, a linear approximation is used. The constants a and b are approximated by VCC for the best fit.

$$\frac{1}{a \cdot SOC + b} \simeq \frac{R_i}{U_{oc}(SOC)^2} \tag{2.17}$$

The battery model can also be simplified with piecewise linear functions to avoid solving an equation of second order. The model assumes a constant voltage of the battery, therefore the loss will only depend on electrochemical power.

$$P_{batt,loss} = \max_{k}(A^k \cdot P_{ech} + B^k) \tag{2.18}$$

2.2.2 Component constraints

The engine, the electric machines and the battery all have limits in terms of power output. All of these are approximated as piecewise linear functions with different number of lines depending on the component. Calculating the correct limits for every operating point are done prior to running the optimisation, implemented in a previous thesis [14] and not further looked into.

2.2.3 Vehicle model

Using the component models from the previous section, the vehicle model can be summed up with four equations. The output power from the driveline is modelled to always exceed the requested power to propel the vehicle, P_{req}, Equation 2.19. This is due to when a large negative request is given from large deceleration, all energy cannot be used for regenerative braking. A variable for the power from using the normal brakes could be introduced, however since this heavy deceleration happens very seldom, this is not seen as necessary.

Equation 2.20 describes how the electrochemical power from the battery is consumed and Equation 2.21 how the power from fuel is used in the ICE.

$$P_{req} \geq P_{ICE,mech} + P_{ISG,mech} + P_{ERAD,mech} - P_{GB,loss} - P_{EGB,loss} \tag{2.19}$$

$$P_{ech} = P_{batt,loss} + P_{ISG,mech} + P_{ISG,loss} + P_{ERAD,mech} + P_{ERAD,loss} + P_{AUX} \tag{2.20}$$

$$P_f = P_{ICE,mech} + P_{ICE,loss} \tag{2.21}$$

The change of the SOC for battery is calculated using Euler forward. E_{scale} represents the transformation from energy in the battery in Joule to SOC, where Q_0 is the charge of the battery.

$$P_{ech}(t) = \frac{E_{scale}}{\Delta t}(SOC(t) - SOC(t+1)) \tag{2.22}$$

$$E_{scale} = Q_0 \cdot U_{oc} \tag{2.23}$$

The requested power to propel the vehicle, P_{req}, is based on calculating the required wheel torque, as Equation 2.24. The different parts that contribute to the required torque is seen in Equation 2.25. The road load polynomial $f(v_{vehicle})$ describes rolling- and air resistance. The inclination of the road, θ, as well as the acceleration of the vehicle affects the required torque.

$$P_{req} = Tq_{req} \cdot \omega_{wh} \tag{2.24}$$

$$\begin{split} Tq_{req} = r_{wh} \cdot v_{vehicle} \cdot f(v_{vehicle}) + sin(\theta) \cdot m_{vehicle} \cdot g + \\ a_{vehicle}(m_{vehicle} + J/r_{wh}) \end{split} \tag{2.25}$$

2.2.4 Operating modes

A usage of operating modes is introduced in last year's thesis [14] to describe how the driveline can be configured for different propulsion methods. The ERAD can either be engaged or not using the EGB. The combustion engine can either be turned on or off. Since the ISG is directly connected to the ICE, a mutual variable is created for these. The 8-speed gearbox can have gear 1 to 8 and the neutral. This results in a total 36 (2*2*9) combinations of modes. When the ICE is turned off, the choice of gear is not relevant and can be set to the neutral. Due to that, the number of modes is reduced to 20, presented in Table 2.1.

Table 2.1: *The complete set of modes for the vehicle.*

Mode	ERAD	ICE+ISG	Gear
1	off	off	0
2	off	on	0
3	off	on	1
4	off	on	2
5	off	on	3
6	off	on	4
7	off	on	5
8	off	on	6
9	off	on	7
10	off	on	8
11	on	off	0
12	on	on	0
13	on	on	1
14	on	on	2
15	on	on	3
16	on	on	4
17	on	on	5
18	on	on	6
19	on	on	7
20	on	on	8

Logic control of the ERAD

The thesis [14] investigated controlling the ERAD and the EGB clutch with logic. In that case the speed and torque demand defines if the ERAD should be engaged or not. The set of rules are given in VSIM and not further investigated. To control the ERAD and EGB clutch with logic reduces the operational modes seen in Table 2.1 to the set of operational modes seen in Table 2.2.

Table 2.2: The set of modes when using logic control of the ERAD.

Mode	ICE+ISG	Gear
1	off	0
2	on	0
3	on	1
4	on	2
5	on	3
6	on	4
7	on	5
8	on	6
9	on	7
10	on	8

3 Optimisation

This chapter presents the theory behind the optimisation methods used in the approach to design the optimisation algorithm. The theory behind the methods is explained and in some cases exemplified. The chosen solvers for the methods are also in some cases evaluated against others, where the choice of the solver is explained.

3.1 Dynamic Programming

DP is a very common method for offline optimisation, as presented in Section 1.3. The method is a way of investigating and storing the cost for all possible combinations to a specific problem. By finding the path with the lowest cost, a global optimal solution is found. To further explain the functionality, the problem analysed in this thesis will be used as an example. To solve the problem, a two-dimensional grid is used. One axis represents the number of time steps or segments and the other the set of modes, explained in Section 2.2.4. The number of nodes in the grid correspond to the size of the problem. A simplified problem with 3 modes and 4 time steps is illustrated in Figure 3.1. N is for this example equal to the number of segments. The numbers in the nodes shows the indices and not the cost.

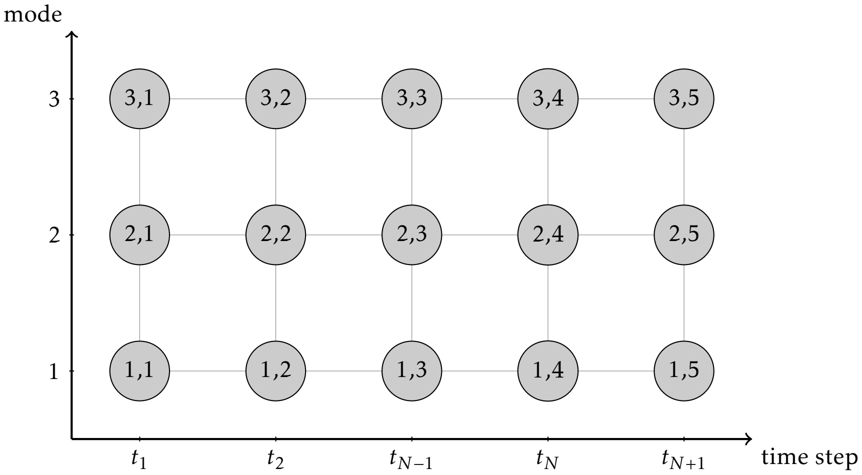


Figure 3.1: *An empty grid for a DP problem with 3 modes and 4 time steps. The last column of nodes is in this setup only used to define the final time and SOC for the battery.*

The method works by calculating the cost for the current node, described as the subproblem, starting from the second to last column in the grid. Figure 3.2 shows the first step for the nodes of mode 1 and 2. Since no costs are present in the last column, only the cost for running with the current mode from t_N to t_{N+1} is added to the total cost.

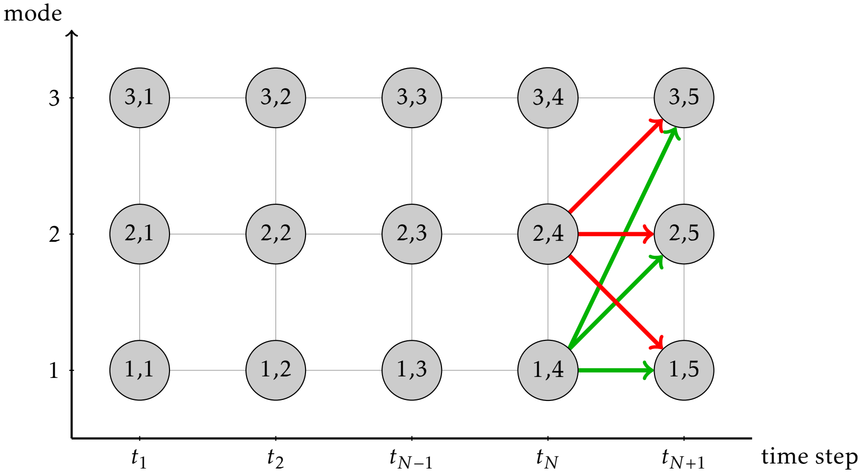


Figure 3.2: *The first step of the DP algorithm. All combinations of paths to the next node are evaluated and the one with the lowest cost is chosen. The green arrows represent all possible subproblems to solve for the first node*

The subproblem for a vehicle is to minimise the fuel consumption in each time step. When using a hybrid vehicle, the battery and electric machines add another degree of freedom. ECMS can be used to solve the subproblem, see Section 3.2.

For the next column of nodes the subproblem is solved and the cost added to the cost for all subsequent nodes. The chosen node to go to is where the cost is lowest, illustred in Figure 3.3 in blue for the first step.

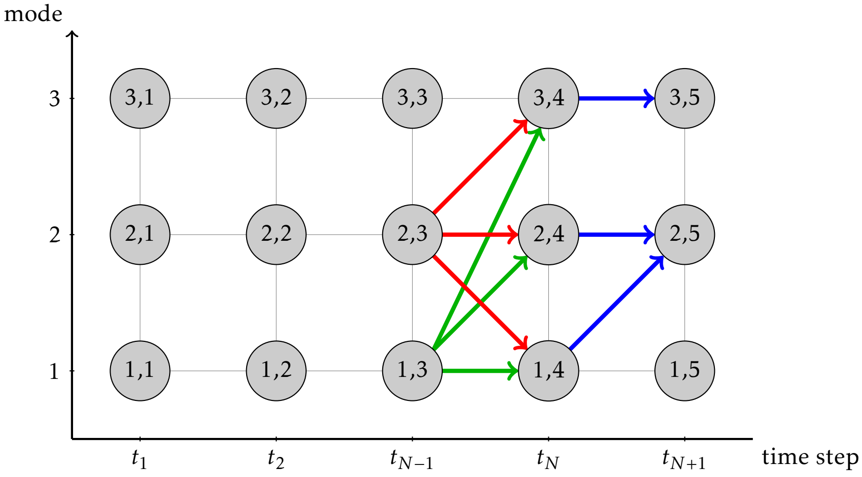


Figure 3.3: *The second step of the DP algorithm. The cost of a node in the N-1th column is added to each succeeding, shown as green arrows for the node for the first mode. The combination with the lowest total cost is saved as seen in blue for the first step.*

The process is repeated for all columns and the local solution with the lowest cost is stored. To find the optimal solution, the nodes in the first column stores the accumulated cost when starting there. By choosing the starting node with the lowest accumulated cost, the global optimum is found. The path through the grid, illustrated as an orange line in Figure 3.4, shows the optimal choice of modes for each time step.

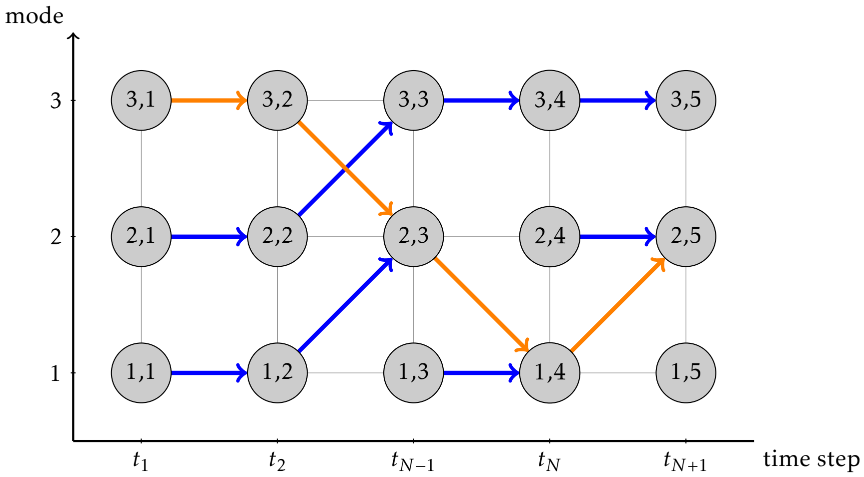


Figure 3.4: *The final grid from the DP algorithm. The blue lines represent the cheapest cost from each node and the path with the lowest total cost is extracted and shown in orange colour.*

3.2 Equivalent Consumption Minimisation Strategy

ECMS is an optimisation strategy directly comparable to PMP. In PMP the Hamiltonian is used, an equation to describe the optimisation to solve, Equation 3.1. The optimal control signal u* is when the Hamiltonian is minimised.

$$u^* = argmin\, H(x(t), u(t), \lambda(t)) \tag{3.1}$$

For a hybrid vehicle, the energy consumption can be from both fuel energy and electric energy. ECMS introduces the equivalence factor, λ in the Hamiltonian, see Equation 3.2. Since the fuel and electrochemical power cannot be directly compared, the equivalence factor can be described as the price for using electrical energy.

$$H = P_f + \lambda \cdot P_{ech} \tag{3.2}$$

The Hamiltonian is minimised in each time step to find the optimal torque distribution to propel the vehicle.

$$[T_{ICE}, T_{ISG}, T_{ERAD}] = argmin\, H \tag{3.3}$$

The strength with this method is that there exists a specific value of the equiva-

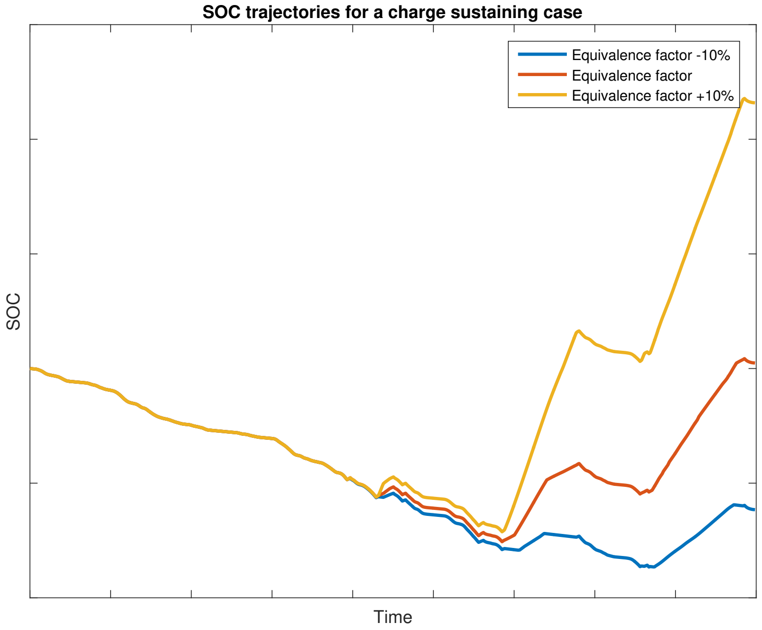


Figure 3.5: *The SOC trajectory for a charge sustaining case. The equivalence factor λ is increased and decreased by 10 percent to test the sensitivity.*

lence factor if the drive mission is known. That is used in [14], where different values of the equivalence factor is tested to find the one that best fulfils the constraints on the final value of the SOC. If the correct equivalence factor is found, that implies ECMS have found the optimal solution to the problem, however the lower and upper constraints of SOC can be violated during the SOC trajectory.

The voltage of the battery varies with the charge level, as described in Section 2.2.1. This results in a lambda that varies over time and is hard to identify accurately from guessing. Using the wrong value of λ can result in a SOC trajectory that varies significantly from the desired one. A change of 10 percent from the optimal value result in large differences in the trajectories as seen in Figure 3.5.

3.3 Convex Optimisation

Convex optimisation can as discussed in Section 1.3 be used with good results when connected with other algorithms such as DP. A convex optimisation problem is a problem on the form as in equation 3.4, e.g., see [20].

$$\begin{aligned} &\text{minimise} && f_0(x) \\ &\text{subject to} && f_i(x) \leq b_i, \ i = 1, \dots, m. \end{aligned} \tag{3.4}$$

where the functions $f_0, \dots, f_m : \mathbf{R}^n \rightarrow \mathbf{R}$ are convex, meaning that they satisfy

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y)$$

for all $x, y \in \mathbf{R}^n$ and all $\alpha, \beta \in \mathbf{R}^n$ with $\alpha + \beta = 1, \alpha \geq 0, \beta \geq 0$[20].

3.3.1 Solvers

There are many different solvers commonly used for solving convex optimisation problems. Some commonly used solvers are SeDuMi and SDPT3, both available from the framework that is CVX, a framework for modelling and describing problems using MATLAB [21]. CVX might be quite good in developmental stages, as one tries to define a problem and solve it. However as the time for CVX to setup a problem and describe it using the specified framework is long, it does not perform as well for problems that needs to be setup many times with different variables. CVX is a commercial solver [21].

For implementations where time is a factor, there are solvers made for being embedded. One of those is CVXGEN, a solver which optimises offline to generate a solver that works very well for small problems and can be used to solve those in a very small amount of time. It generates a fast custom solver for the specified problem [22]. However, to generate a solver for a specific problem gives limitations in the problem setup. If the problem changes, the solver has to change. Therefore, this makes CVXGEN hard to use for purposes when the problem formulation changes. CVXGEN is a commercial solver [22].

Another of the solvers made for embedded purposes is the Embedded Conic Solver (ECOS). It is a convex optimisation solver which can handle second order cone programs (SOCP). It is written in ANSI C, with about 1000 rows of code, making it very fast and efficient. It utilises a very fast converging interior point algorithm for solving the problem [23]. The layout of ECOS enables utilisation of the solver for very fast problem setup and solve. It is therefore compatible to use when the problem layout is changing. ECOS is an open source solver, distributed under the GNU General Public License v3.0 [24].

3.3.2 The dual problem

When solving an optimisation problem, the problem at hand is called the primal problem, which can be exemplified as in Equation 3.5. When a primal problem exists, there is always a corresponding dual problem. For the given linear example in Equation 3.5 the dual problem is formulated as in Equation 3.6, where μ is the dual variable. There exists a dual variable for each constraint[25].

$$\begin{aligned} \text{minimize} \quad & max\ c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{3.5}$$

$$\begin{aligned} \text{maximise} \quad & max\ b^T \mu \\ \text{subject to} \quad & A^T \mu \geq c \\ & \mu \geq 0 \end{aligned} \tag{3.6}$$

If the primal problem does have an allowed, limited optimal solution, x^*, then the dual problem does also have an allowed, limited optimal solution, μ^*[25].

When using convex optimisation on a torque split problem for a HEV, the dual variable for the SOC constraints, in this case called λ will be equivalent to the equivalence factor λ used in the Hamiltonian in ECMS, from Section 3.2[7]. This fact is the key behind how an efficient solver can be designed. By solving a convex optimisation problem, the resulting dual variable for the SOC constraints gives information about how the ECMS-problem can be solved.

4

Segmentation

With the recent advances in autonomous functions for vehicles, the navigational system has had to be updated. Therefore nowadays the navigational systems for cars and other vehicles have become quite advanced with functions like ADASIS included. Those functions can be used to help plan the trip, in terms of how the energy management in the vehicle should act during the trip. With the input of a destination from the user of the vehicle, the navigational system of the car is able to output data, using ADASIS output horizon data containing, among others, speed limits and altitude profiles. This data can then be used to plan the trip. In this thesis, the data is used in the vehicle EMS. However, the data that comes straight from ADASIS has a low resolution but can still result in a lot of information, meaning that to optimise directly from that data would be time consuming. Therefore, this chapter will examine an approach for a segmentation algorithm. The algorithm should be able to merge data from the ADASIS output to larger segments, with length up to a couple of kilometres, to limit the amount of data to process in the optimisation algorithm. An illustration of how the segmentation could work can be seen in Figure 4.1.

The improvements that can be made with segmentation can be seen in for example [26] where a development of ECMS using segmentation of data from the navigation system was covered. The reference signal in SOC was modified given the route ahead and major improvements compared to a non-predicting ECMS is shown. A segmentational approach is also done by the authors of [12], where the road is segmented and where the segments are later sorted into order by the power demand for each segment. The optimisation is then done on the sorted segments. This strategy were successful and could lead to decreases in fuel consumption and the time when the engine is running [12].

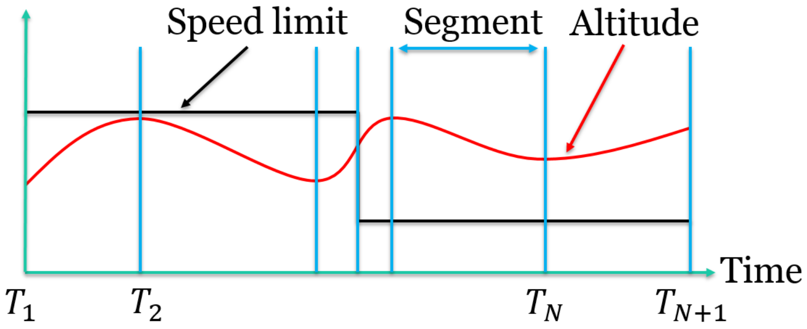


Figure 4.1: *Illustration of how data about altitude and speed limit can be segmented.*

4.1 ADASIS

ADASIS is a protocol in which the communication interface from the navigational system to the rest of the car is specified. The communication is done over the Control Area Network (CAN). The information provided is an Advanced Driver-Assistance Systems (ADAS) horizon, which allows the vehicle to gain knowledge of the road forward, further than ordinary sensors would allow as the data used is from the navigational system. ADASIS is widely used and developed by a coordinator that partners with many of the major automotive companies[27].

There are two different versions av ADASIS, one version for short range applications and one version for long-range applications. The difference between the two versions is the sampling rate of the data. The short-range version gives information about the route very often, while the long range version gives information about the route with longer distance between the points. The version used in this thesis will be the long-range version, as the routes that are used will be of substantial length. The data collection from ADASIS and the basic functionality in making that data useful is done in the same way as the authors from [12] did.

4.2 Data

As the data from long-range ADASIS is sparser, it is not that precise. Together with the fact from what the authors of [28] found, that a driver seldom drives according to the speed limit of the road, the data a segmentation algorithm will take decisions on will have faults. The same authors did also find that adapting a general driver model is very difficult [28]. With those limitations, segmentation will not be able to provide a perfect approximation to the route.

The number of segments does also matter. According to [29] there is a limit for when a substantial improvement is seen for each added segment.

5 Method

To construct a route based optimal EMS for a HEV, there are many factors that have to be taken into account when optimising. The main factors vehicle factors are discrete decisions such as gear choice and combustion engine on/off, torque split and the available energy. Other main factors are environmental or route factors such as the road profile and speed limits. By combining several of the methods presented in earlier chapters, a computational efficient algorithm can be constructed while keeping good results. All implementation is done in MATLAB.

5.1 General description

This thesis presents a method where DDP and ECMS is combined with convex optimisation and a segmentation algorithm, resulting in a SOC reference trajectory. The algorithm for discrete decisions using DDP and ECMS was developed in a previous thesis [14] and is further developed to decrease the computational time required for each iteration. Previously, the algorithm used a root-finding algorithm to find the correct value of the equivalence factor for the whole drive cycle, a quite time-consuming process.

In the implementation described in this thesis, the root-finding algorithm is replaced by convex optimisation. As stated in Section 3.3.2, the dual variable for the SOC-constraint in convex optimisation and the equivalence factor in DDP/ECMS is for this problem the same and is hereafter given the common description λ.

One great benefit with convex optimisation is the ability to find the correct λ for the whole drive cycle, even if hitting a constraint. λ is therefore not set to one value but can change during the drive cycle. The convex optimisation is implemented with the Embedded conic solver ECOS.

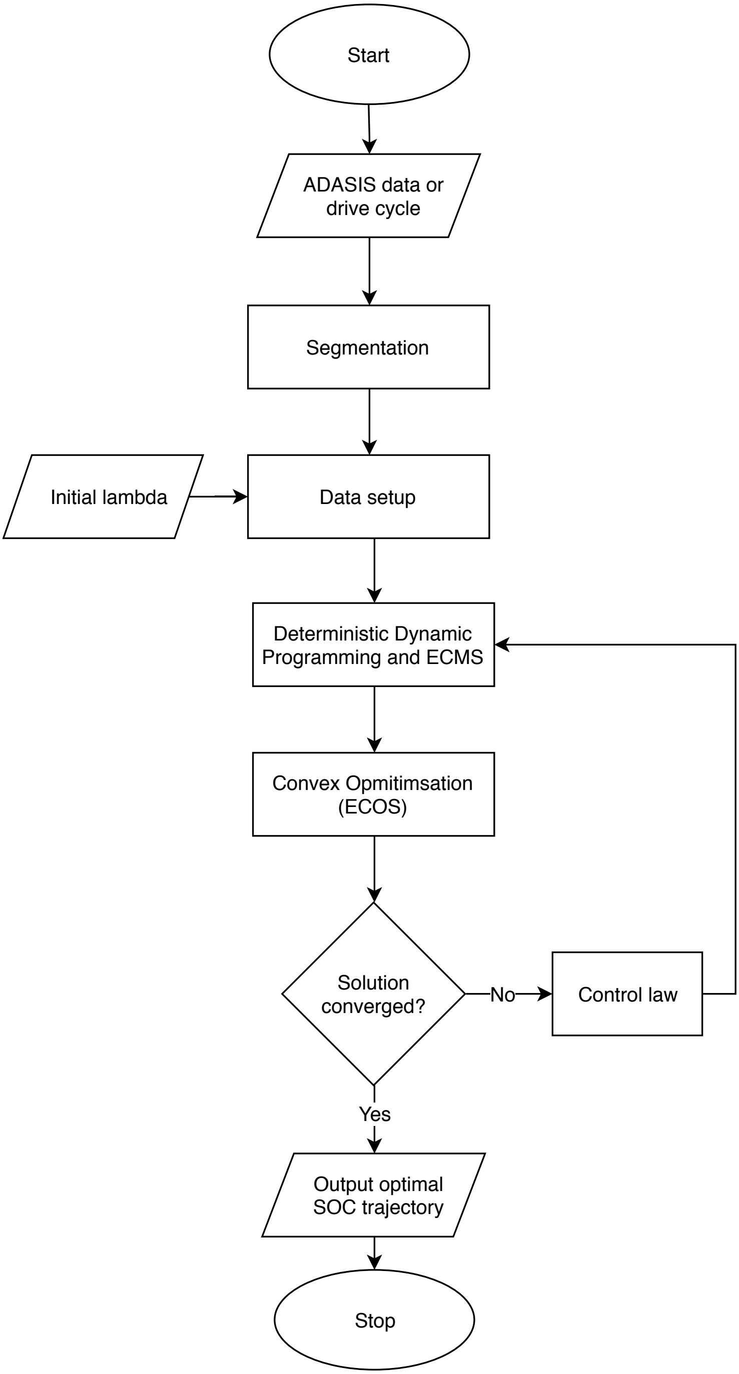

Start
ADASIS data or drive cycle
Segmentation
Initial lambda
Data setup
Deterministic Dynamic Programming and ECMS
Convex Opmitimsation (ECOS)
Solution converged?
No
Control law
Yes
Output optimal SOC trajectory
Stop

Figure 5.1: *Flow chart of the complete algorithm presented in the method*

The implemented segmentation algorithm starts with data from the navigational system through ADASIS, which is segmented according to speed limits and the road inclination. The segments are then used as input to the optimisation algorithm, efficiently limiting the amount of data to the algorithm while keeping a high quality to maintain accurate results. This makes it possible to run the optimisation algorithm in a small amount of time.

The complete optimisation algorithm using DDP and ECMS combined with convex programming requires handling of how λ is updated between the solvers. λ has to be guessed to start the solver, which can be problematic as a guess far away from the actual value might cause the solver to be unable to start iterating. An approach to handle this is presented in Section 5.6. A flow chart of the complete algorithm is presented in figure 5.1.

For validation in an online environment, Volvo Simulation Tool VSIM is used. To be able to test the algorithm, a way to control the SOC of the battery is required. This controller is constructed in VSIM and shall aim to follow the reference SOC trajectory from the algorithm.

5.2 Segmentation

The segmentation is made using data on road inclination and speed restrictions to produce segments of the drive cycle.

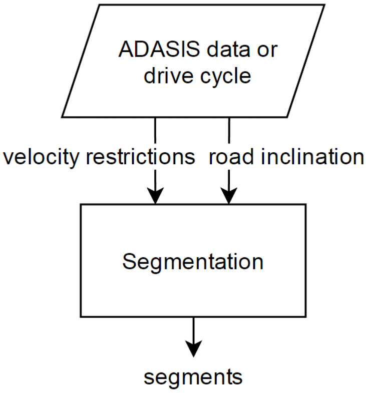


Figure 5.2: *Input and output for the segmentation algorithm.*

5.2.1 Data

The data used for input into the algorithm comes from ADASIS and is recorded during VCC test drives. The collected data is then processed, and the different routes sent from ADASIS is used. ADASIS sends a new route for every recalculation of the route that it does, meaning that when the driver deviates from the given route, ADASIS recalculates a new route.

Those routes are then used to generate drive cycle data, to convert the data to something that can later be usable in the optimisation algorithm. The data not in routes are sorted out and the routes with to small amount of data is thrown away. The connection of the altitude points is first done using a linear approach connecting the different points, thereafter the linear approach is interpolated with regards to the time, so that each time point holds a data point.

5.2.2 Algorithm

Input

The algorithm will use data on speed limits and altitude data from ADASIS to make decisions about when to start or end a segment. In developmental purposes, other drive cycles will be used as well, as long as they contain the appropriate data.

Computations

As the algorithm needs to be fast it is strictly using logic. The logic based choices are made on the speed limits, the altitude change between the segments and the first and second derivative of the altitude. There are however some overriding logic. No segmentation is made if the distance in which a new speed limit is valid is too short. There is also a minimum distance specified between segments.

The overriding logic is necessary for eliminating small changes that would make a segmentation with the original logic, which would greatly increase the number of segments and lead to no or almost no improvement.

Output

The segmentational algorithm provides the number of segments, the time when the different segments are active, the altitude data, the speed limits and the road inclination. To be able to closer examine how the implemented segmentation will work in the finished algorithm, recorded speed data will be used to provide the segments with data of the mean speed during each segment and the standard deviation of the speed from mean in each segment.

The standard deviation is found using the Matlab implementation of Equation 5.1, where σ is the standard deviation, n is the amount of samples, X_i is the value in each sample point and $\overline{X}$ is the mean over all samples.

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^2} \tag{5.1}$$

The implementation with standard deviation doubles the number of segments, with half of each segment implemented as the standard deviation over the segment subtracted to the mean of the segment and the other half instead with the standard deviation added to the mean of the segment.

5.3 Data setup

Before running the solver algorithms, data about the problem and the vehicle is loaded. The maps that describe losses are precomputed for all modes and time steps. Due to DDP testing all possible combinations in every iteration, this method is very efficient in terms of saving time. By only storing the data that is relevant, the memory usage can be kept small.

The maps of losses in each component are only specified for a few angular velocities, as presented in section 2.2.1. Interpolation is required to find the correct piecewise linear functions for a given angular velocity, where a faster method compared to the standard in Matlab is developed.

5.3.1 Operating mode reduction

In Section 2.2.4, the number of modes when controlling the ERAD based on logic was determined to be ten. This thesis introduces a way of reducing the number of modes tested in each segment by selecting a maximum of three suitable gears depending on the speed of the vehicle. For high velocities there is no relevance evaluating the lower gears and there is no need evaluating the highest gears when the vehicle speed is low.

The method is based on a map for gear selection provided by VCC. The most suitable gear for each segment is calculated with the nearest neighbour method, called $gear_{suggestion}$. To use the potential of DDP to test multiple alternatives, plus-minus one gear is added to the set of gears, $gear_{set}$ if possible.

$$gear_{set} = \begin{cases} \{0\} & gear_{suggestion} = 0 \\ \{1, 2\} & gear_{suggestion} = 1 \\ \{gear_{suggestion} \pm 1\} & gear_{suggestion} \in [2, 7] \\ \{7, 8\} & gear_{suggestion} = 8 \end{cases} \tag{5.2}$$

When the ERAD is controlled through optimisation, the reduction of operating modes will not be used. This is to test how a maximum 20 set of operating modes compares to a greatly reduced number when controlling the ERAD through logic.

This reduction of operational modes is hereafter referred to as gear reduction.

5.3.2 Acceleration between segments

The data from the segmentation algorithm contains segments with constant velocity and inclination. Changing the velocity of the vehicle between segments is something being ignored with this simplified data. To account for that, a method of adding segments with constant acceleration between each segment where the velocity changes is tested. The velocity in these segments is set as the mean value of the starting and final velocity. A constant acceleration is used, which gives a segment length depending on the change in velocity. The new segment starts where the velocity profile changes, i.e. where the speed limit sign is placed.

5.4 Dynamic programming and ECMS

The first part of the optimisation algorithm consist of optimising the discrete variables, such as choice of gear and engine on/off, for all segments. For this, a DDP-approach is chosen. The method will find the global optimum and is rather efficient when the number of optimisation variables are limited. ECMS is used to get the torque split for each segment in order to get the cost for using a specific operating mode. The script for DDP and integration of ECMS origins from the thesis[14] and has been improved in this thesis.

The input to the solver is data about route and and vehicle data for all operating modes and segments, as well as the λ for all segments. For the first iteration λ has to be initialised.

The output is the choice of operating mode for all segments, as shown in Table 2.2. That includes choice of gear and engine on/off for the combustion engine. If the ERAD is not controlled with logic, the output needs to include the status of the ERAD as well, as can be seen in Table 2.1.

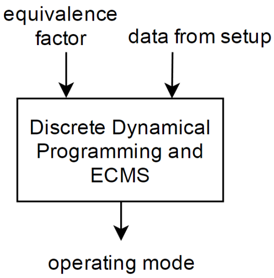


Figure 5.3: *Input and output for the DDP/ECMS algorithm.*

5.5 Convex optimisation, ECOS implementation

All problems solved by the convex solver ECOS needs to be on matrix form. The problem set up is as presented in Equation 5.3.

$$\begin{aligned} \text{minimise} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & Gx \preceq_K h \end{aligned} \tag{5.3}$$

Where, as specified in [24], the symbol $\preceq_K$ denotes generalised inequality with respect to the cone **K**. From now, let **K** be the second order cone, i.e. the vector $h - Gx$ belongs to the second order cone **K**:

$$Gx \preceq_K h \Leftrightarrow s = h - Gx \in K$$

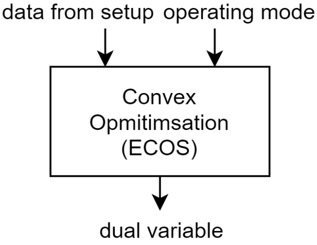


Figure 5.4: *Input and output for the convex optimisation.*

Since the studied configuration as specified in Figure 2.5, can be used with different ways of propulsion, four cases have been made for these. Which engine is active or not active in which case can be found in Table 5.1. The cases can be directly linked to the modes as specified in Table 2.1. As the cases are different in use of drivline components, there is no idea of solving a variable related to a component that is not active.

Table 5.1: *Description of different cases in ECOS.*

	Case 1	**Case 2**	**Case 3**	**Case 4**
ICE	On	Off	On	Off
ERAD	On	Off	Off	On

This result in the number of variables in the problem formulation to change depending on which case is present in each time step. The variables for the four cases can be seen in Table 5.2. Therefore, the problem formulation part iterates the build of matrices over the number of time steps in the drive cycle. For each time step, one case is active, and for that case the correct variables are put into the problem matrices.

Table 5.2: The variables during different cases in ECOS described in Table 5.1. Where $Slack$ is introduced in Equation 5.8.

Var.nr	**Case 1**	**Case 2**	**Case 3**	**Case 4**
1	P_{ICE}	SOC	P_{ICE}	P_{ERAD}
2	$P_{ICE,loss}$	$P_{Batt,loss}$	$P_{ICE,loss}$	$P_{ERAD,loss}$
3	P_{ISG}	P_{ech}	P_{ISG}	P_{EGB}
4	$P_{ISG,loss}$	$Slack$	$P_{ISG,loss}$	SOC
5	P_{ERAD}		P_{GB}	$P_{Batt,loss}$
6	$P_{ERAD,loss}$		P_f	P_{ech}
7	P_{GB}		SOC	$Slack$
8	P_{EGB}		$P_{Batt,loss}$	
9	P_f		P_{ech}	
10	SOC		$Slack$	
11	$P_{Batt,loss}$			
12	P_{ech}			
13	$Slack$			

5.5.1 The objective function

The main goal with the optimisation is to minimise the fuel consumption over all segments. The fuel energy is calculated using the variable for fuel power P_f. Since all dynamics are removed from the models, the fuel energy consumed over one segment, Q_{fuel}, can be seen as the fuel power times the length of the segment, Equation 5.4

$$Q_{fuel} = P_f \cdot \Delta t \tag{5.4}$$

This gives the objective function as Equation 5.5, where t represents one segment and t_N is the last segment.

$$\text{minimise} \sum_{t=1}^{t_N} Q_{fuel}^t \tag{5.5}$$

If the requested regenerative breaking power exceeds the maximal power that either the battery or the ERAD and ISG together can recover, the difference has to be accounted for in the model. To avoid the solver from using the gearboxes to recover that energy, which is not possible in reality, a minor minimisation is used on these as well.

As the number of active components differs with the actual case, the objective function depends on this and one objective function for each case is formulated. The objective functions for the different cases can be seen in Equation 5.6. The final objective function will be a combination of these, depending on the cases in the problem to solve.

$$\begin{cases} \text{minimise} \sum_{t=1}^{t_N} Q^t_{fuel} + s \cdot P^t_{GB} + s \cdot P^t_{EGB} & case = 1 \\ \text{minimise} \sum_{t=1}^{t_N} s \cdot P^t_{ech} & case = 2 \\ \text{minimise} \sum_{t=1}^{t_N} Q^t_{fuel} + s \cdot P^t_{GB} & case = 3 \\ \text{minimise} \sum_{t=1}^{t_N} s \cdot P^t_{EGB} & case = 4 \end{cases} \tag{5.6}$$

Where s is the weight and is equal to 0.001 for all cases.

5.5.2 Models

The problem to solve will depend on the case that is active. From Table 5.2, the problem at hand for each case can be seen. The inequality and equality models used are described in Sections 2.2.1 and 2.2.3, with the exception of the implementation of the battery as a second order cone, which is described below.

Second Order Cone Programming

ECOS does have the ability to solve problems where second order cones are included. This ability is used as the quadratic over linear battery model as seen when combining Equations 2.16 and 2.17 can be formulated as a second order cone. The quadratic over linear function is convex if the denominator is greater than zero[20], thus it can be used in the convex optimisation.

To make an easy conversion, the program QCML [30] is used to convert the quadratic over linear equation to a second order cone. For the input arguments to ECOS, the conversion is as follows: from Equation: 5.7 to Equations: 5.8 and 5.9. Where the matrix in Equation 5.8 describes the linear relationships, the two first rows for the linear constraints and the last row for the cone and the matrix in Equation 5.9 describes the quadratic relationship as a second order cone **K**.

$$\begin{aligned}
&P_{batt,loss} = \frac{1}{f_1(SOC)} \cdot {P_{ech}}^2 \\
&P_{ech} - P_{batt,loss} \leq +f_2(SOC) \\
&P_{ech} - P_{batt,loss} \geq +f_3(SOC) \\
&SOC \geq 0
\end{aligned} \tag{5.7}$$

Where $f_1(SOC) = a_1 \cdot SOC + b_1$ comes from a linear fitting of $\frac{R_i}{U_{oc}(SOC)^2}$ in the battery model described in Equation 2.16. And where $f_2(SOC) = a_2 \cdot SOC + b_2$ and $f_3(SOC) = a_3 \cdot SOC + b_3$ comes from linear fittings of maximum respective minimum limits of the output power from the battery, $P_{ech} - P_{batt,loss}$, depending on the SOC.

$$\begin{bmatrix} -a_2 & -1 & 1 & 0 \\ a_3 & 1 & -1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} SOC \\ P_{Batt,loss} \\ P_{ech} \\ Slack \end{bmatrix} \leq \begin{bmatrix} b_2 \\ -b_3 \\ 0 \end{bmatrix} \tag{5.8}$$

$$\begin{bmatrix} -a_1 & 0 & 0 & -1 \\ -a_1 & 0 & 0 & 1 \\ 0 & 0 & -2 & 0 \end{bmatrix} \begin{bmatrix} SOC \\ P_{Batt,loss} \\ P_{ech} \\ Slack \end{bmatrix} \preceq_K \begin{bmatrix} b_1 \\ b_1 \\ 0 \end{bmatrix} \tag{5.9}$$

Where a_1, b_1, a_2, b_2, a_3 and b_3 in Equation 5.9 comes from $f_1(SOC)$, $f_2(SOC)$ and $f_3(SOC)$ in Equation 5.7 and where **K** is a second order cone.

Scaling

To achieve accurate results from the convex solver, all variables to be solved should be of around an equal size. Since SOC is of magnitude 10^0 and P_{fuel} of magnitude up to 10^6, the solver will run into numerical problems if the scaling is not done properly. In the implementation for the thesis, the scaling used results in variables with a value in the interval $[-1, 1]$. A diagonal scaling matrix α with the same dimensions as the number of variables is multiplied to the problem setup matrices. Each diagonal element represent the scaling being used for the corresponding variable.

$$\begin{aligned}
\text{minimise} \quad & (c\alpha)^T x \\
\text{subject to} \quad & (A\alpha)x = b \\
& (G\alpha)x \preceq_K h
\end{aligned} \tag{5.10}$$

5.6 Connecting the algorithms

To connect the algorithms the operating modes being calculated by DDP/ECMS is used in the convex optimisation to calculate the optimal λ. The connection is illustrated in Figure 5.5. λ_{in} denotes the value being sent to DDP/ECMS and λ_{out} the value from ECOS. The initial λ has to be guessed. By updating λ_{in} for the next iteration based on λ_{out}, the result from both solvers should converge.

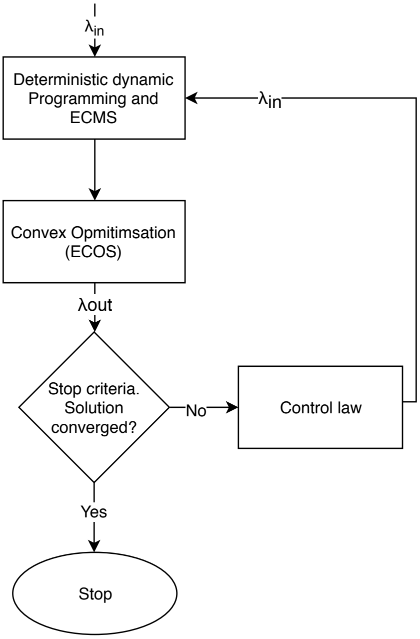

λin
Deterministic dynamic Programming and ECMS
λin
Convex Opmitimsation (ECOS)
λout
Stop criteria. Solution converged?
No
Control law
Yes
Stop

Figure 5.5: *Illustration of how the optimisation algorithms are connected and how λ is updated between iterations.*

5.6.1 Control law

The key behind solving the problem quickly is to update λ in a smart way to ensure that all kinds of drive missions are solved fast and accurate. The authors of [7], suggest a method where a weight factor q decides how to update λ. This method is seen as easy to understand and is used to update λ to the next iteration. As seen in Equation 5.11, the weight factor decides how much to trust the difference between the solvers for the next iteration $i + 1$. This can be described as a factor of how accurate the solution from the convex solver is. The weight factor varies between 0 and 1, where a higher value means that the new value from the convex solver should be trusted. For the extreme case where q is zero, no regards is taken to λ_{out} from the convex solver and the previous λ_{in} is used again. This will not change anything between iterations and is considered as a non-existing case. Conversely, if q is equal to one, λ_{out} from the convex solver is directly used in the next iteration.

$$\lambda_{in}{}^{i+1} = \lambda_{in}{}^{i} + q \cdot (\lambda_{out}{}^{i} - \lambda_{in}{}^{i}) \tag{5.11}$$

To give a value of how close to the correct solution the algorithms are, the root-mean-square error is used. The value is calculated based on the sum of the square of the error in each time step, as seen in Equation 5.12.

$$\begin{aligned} \Delta\lambda(t) &= \lambda_{in}(t) - \lambda_{out}(t) \\ \lambda_{RMS} &= \sqrt{\frac{1}{N_{seg}}\left(\Delta\lambda(1)^2 + \Delta\lambda(2)^2 + \ldots + \Delta\lambda(N_{seg})^2\right)} \end{aligned} \tag{5.12}$$

By updating $q \in]0,1]$, λ converges to optimal where λ_{RMS} is close to zero. The choice of q has to be a trade-off between reducing the computational time and minimising the risk of over/undershooting the value of λ. A flow chart of how q is updated is seen in Figure 5.6

Running the DDP/ECMS algorithm with a too high or low value of λ will result in a large error in the SOC trajectory and a trajectory of operating modes that is not optimal. If then the convex problem is solved with constraints for the start and end point for the SOC are fixed within small limits, the result can be trouble finding a feasible solution. This might also reduce in a bad solution in which the results are wrong due to the convex solver being too constrained to solve the problem in a way so that the optimal λ is found. For example, if the ICE is turned off in many segments due to a low guess of λ, while the drive mission should be charge sustaining, the vehicle cannot generate enough energy to the battery during the few segments the ICE is running to be charge sustaining.

Initial value of the weight factor

The first two iterations needs to be run with a very low weight factor to get closer to the target. This is due to the problem when initialising with a too high or too low λ_{in}, the output value will not be perfect. Depending on the λ_{RMS}-value after the first iterations, q is chosen between 0.4 and 0.6 as seen in Equation 5.13.

$$q = \begin{cases} 0.6, & \lambda_{RMS} \geq 0.3 \\ 0.5, & 0.05 < \lambda_{RMS} < 0.3 \\ 0.4, & \lambda_{RMS} \leq 0.05 \end{cases} \tag{5.13}$$

When controlling the ERAD through optimisation, the λ_{out} can differ significantly from the desired value, although always with a correct sign if λ_{in} should be increased or decreased. To handle this, the weight factor q is given 1/5 of the value in Equation 5.13.

Relaxation of the SOC constraints

An alternative to iterating with a small q and fixed constraints on the start and final value on SOC is to change to a relaxation of these constraints. Two new variables are introduced that describe the SOC trajectory's difference from the desired start and end value. These variables are then added to the objective function to minimise the difference.

$$SOC_{diff,start} = |SOC(t_1) - SOC_{start}| \tag{5.14}$$
$$SOC_{diff,end} = |SOC(t_{N+1}) - SOC_{end}| \tag{5.15}$$

The relaxation solver has the strength of being able to solve the case of where the guessed λ generates a mode trajectory that creates a non-feasible problem. Due to this strength, in the case of controlling the ERAD with logic, the relaxation solver is always used in the first iteration. If the λ_{RMS}-value is between 0.2 and 0.8 after the first iteration, it is used in the second iteration as well. This is to use the potential with the relaxation solver to get closer to the correct value of λ and decrease the number of iterations required with the normal solver.

When the relaxation solver is used, the value of q is set to 1. This means that the resulting λ_{out} is copied directly to be used as input in the next iteration. In the case of optimising the ERAD, the relaxation solver is not used. This is due to that the algorithm in this case is very sensitive to which value λ has.

Decreasing error between iterations

To make sure that the solution converges to the optimal, the λ_{RMS}-value has to decrease between the iterations. If it does not, the λ_{out} from the convex solver has increased or decreased too much. This indicates that the weight factor, q has been too high. The last valid solution where the λ_{RMS}-value was decreasing is stored and to the next iteration q is divided by two.

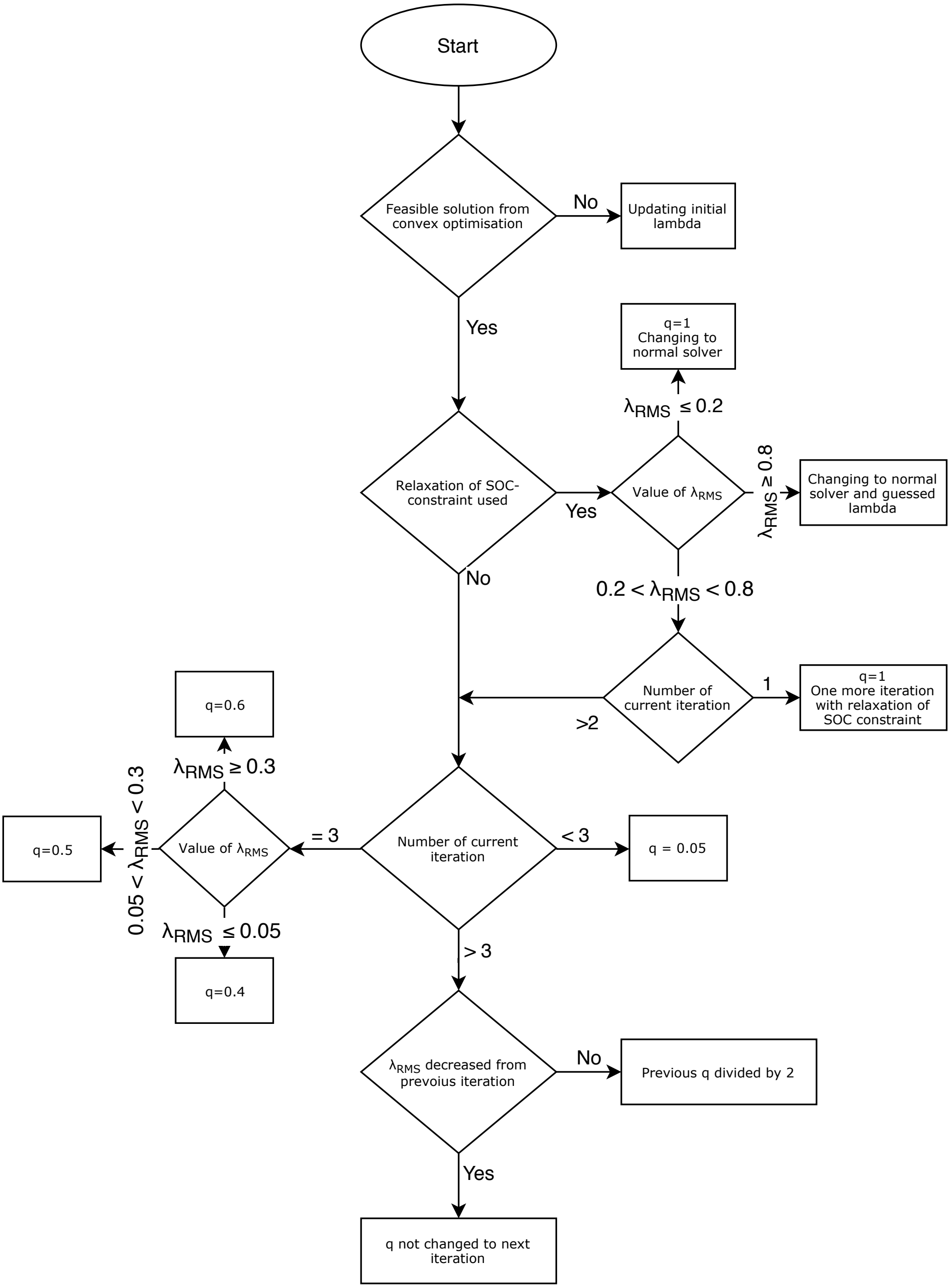

Start
Feasible solution from convex optimisation
No
Updating initial lambda
Yes
q=1
Changing to normal solver
$\lambda_{RMS} \leq 0.2$
Relaxation of SOC-constraint used
Yes
Value of λ_{RMS}
$\lambda_{RMS} \geq 0.8$
Changing to normal solver and guessed lambda
No
$0.2 < \lambda_{RMS} < 0.8$
Number of current iteration
1
q=1
One more iteration with relaxation of SOC constraint
>2
q=0.6
$\lambda_{RMS} \geq 0.3$
q=0.5
$0.05 < \lambda_{RMS} < 0.3$
Value of λ_{RMS}
= 3
Number of current iteration
< 3
q = 0.05
$\lambda_{RMS} \leq 0.05$
q=0.4
> 3
λ_{RMS} decreased from prevoius iteration
No
Previous q divided by 2
Yes
q not changed to next iteration

Figure 5.6: *Flow chart of how the weight factor is updated.*

5.6.2 Stop criteria

Detecting whether the solution converges can be done in several ways. This thesis presents three different alternatives, which combined could be made into many more. The choice of the three alternatives is because each of them tells something important about the problem. The different alternatives is presented in the list below. When any of these criteria are fulfilled, the algorithm will stop. The threshold for the criteria are chosen based on extensive testing of the algorithm with the purpose of avoiding additional numbers of iterations without improvements in the solution.

- λ_{RMS} **below threshold**
- ***q*** **below threshold** If q is very small, there is no further idea to keep iterating as λ won't change enough to affect the result. As the algorithm sometimes cannot find perfect convergence between λ_{in} and λ_{out}, it can however find a value close enough, this gives a solution close to optimal.
- **Combination of several criteria that all have to be fulfilled.** To decrease the number of iterations when close to an optimal solution, multiple criteria are combined. Iterating for λ_{RMS} below threshold is not necessary when no or minor changes are seen between the iterations and the SOC trajectories are similar between the optimisation methods.
    - Minor change of operating mode in the segments between iterations. Suggested in [7] and slightly modified.
    - Change in fuel consumption between iterations below threshold.
    - Difference in SOC trajectory between the DDP/ECMS and convex optimisation below threshold.

Since the relaxation of SOC constraints gives a solution slightly different than the one without, the last iteration has to be with the normal solver without relaxation. The iteration with the lowest λ_{RMS} is stored and retrieved when a stop criterion is reached. The algorithm outputs the optimal SOC trajectory for the problem and the optimal λ for each segments.

5.7 Controller

To evaluate the generated SOC trajectories possibility to save fuel with the more advanced vehicle model in VSIM, a controller is required for the actual SOC to follow the generated trajectory. The objective for the controller is to follow a reference SOC-trajectory, therefore, it does in some way need to be able to control the charge of the battery.

Given the complexity of the model in VSIM, the control method is designed to be simple for a straightforward implementation and validation. The main scope with the thesis is calculating reference trajectories and not designing an advanced controller. Since the reference SOC trajectory is calculated based on segmented road data and a simple vehicle model, the trajectory cannot be perfectly followed. This reduces the need for an advanced controller. Two controllers are designed with the same control method, however with different ways of calculating the reference signal $r(t)$. These two controllers are developed in parallel to test which method that gives best performance. The main idea behind the controllers can be seen in Figure 5.7. The ERAD is not part of the control method.

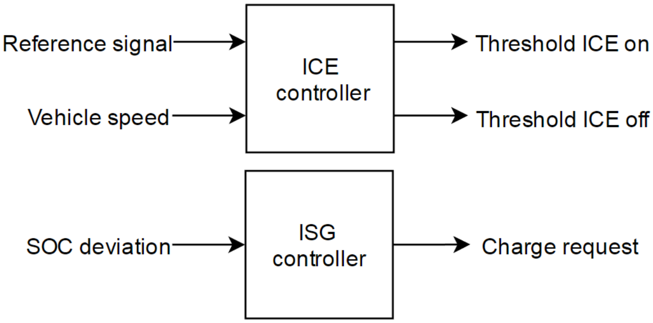


Figure 5.7: *Simple sketch of the input and output of the controller. Only the input to the ISG controller is modified.*

5.7.1 Control method

The control method works by updating the engine thresholds depending on the reference signal. By doing this the SOC reference trajectory can be followed. This works by lowering the engine threshold when the SOC value is below the SOC reference trajectory to force the vehicle to run with the ICE instead of electric propulsion. The opposite holds for when the SOC value is higher than the reference value, in this case to thresholds are increased to force the vehicle to run all electric to consume electrical energy.

The reference signal, $r(t)$, to the controller comes from the two ways of calculating the reference, presented in Section 5.7.2 and 5.7.3. An illustration of the thresholds can be seen in Figure 5.8. The engine is started when the power request is above a threshold, shown in green, and turned off when the request is below a threshold, shown in orange. To reduce the number of engine starts, a band between the lines is used.

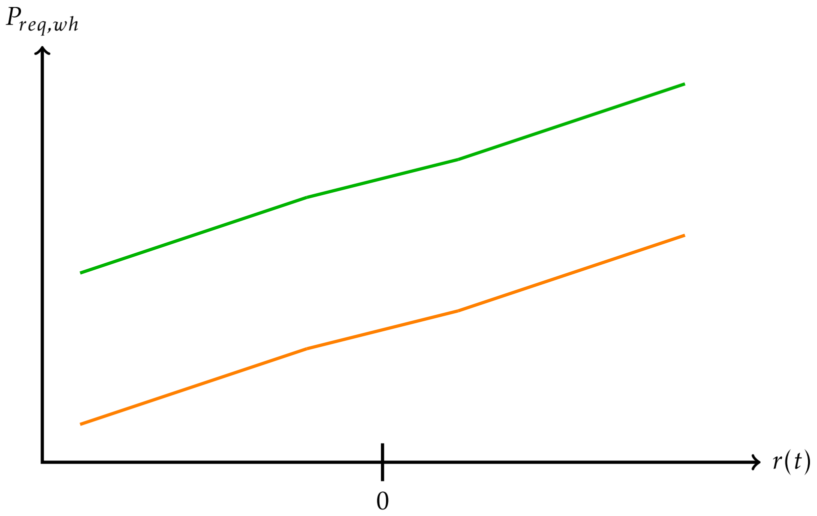


Figure 5.8: *An illustration of how the thresholds for engine on/off is changed depending on the difference from the reference signal $r(t)$.*

The thresholds are specified for a few values of $r(t)$ and velocity of the vehicle. The decision of using the velocity is based on the fact that the vehicle should operate differently depending on the velocity. For a low speed, electrical propulsion is desired, thus increasing the threshold to avoid starting the ICE. For a highway drive, the opposite holds.

Control of charge request

The controller for charge request to the ISG in VSIM is continuously updated with the deviation from the reference SOC trajectory. This controller is based on a number of rules which are not modified.

5.7.2 Deviation from trajectory method

The controller in this method is designed to follow the reference based on the deviation from the reference SOC trajectory in each time step, as seen in Equation 5.16. The engine on/off thresholds are specified for a few deviations from the trajectory, creating bands as seen in Figure 5.9. The further away from the reference trajectory the SOC value is, the more aggressive the controller will be. Interpolation is used to find the correct threshold for the current vehicle speed and deviation.

$$r_{deviationcontroller}(t) = SOC(t) - SOC_{ref}(t) \tag{5.16}$$

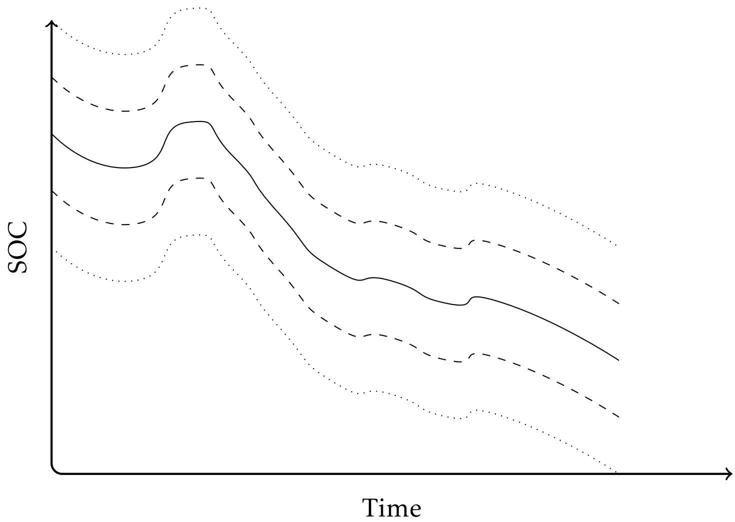


Figure 5.9: An illustration of the principle on how the deviation from reference controller works. The further away from the reference trajectory in solid line, the more aggressive the controller is.

5.7.3 Deviation from gradient method

A second method evaluated is based on the deviation from the gradient to the next point on the SOC reference trajectory, as seen in Equation 5.17. The control parameters are specified for a few deviations from gradient, where a larger difference of the gradient gives a more aggressive controller. Interpolation given the actual deviation in gradient and vehicle speed is used.

$$r_{gradientcontroller}(t) = \frac{SOC_{ref}(t) - SOC(t)}{t_{ref,target} - t} \tag{5.17}$$

Due to the large error in gradient when being close to a target point in time and slightly off in SOC, the next target point is updated when being rather close. The drawback with this method is that peaks in reference are neglected, however it greatly reduces the problem with the controller being too aggressive when it is close to a target.

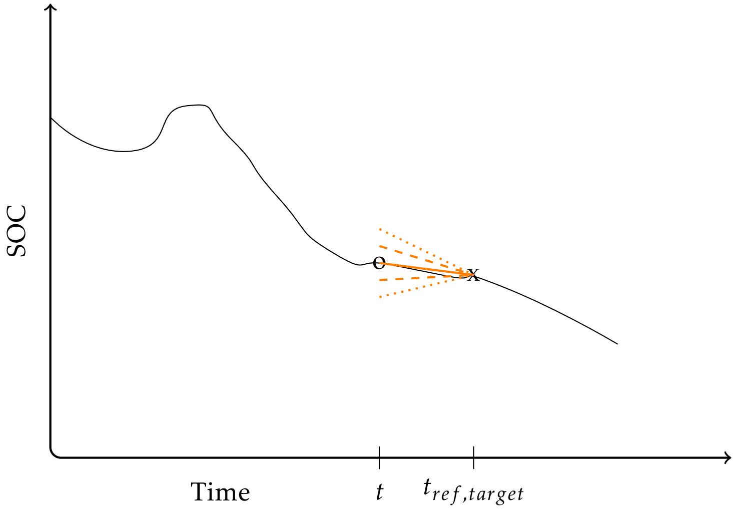


Figure 5.10: An illustration of the deviation from gradient to next target controller. The current target is marked with a x and the reference for the current time marked with o. The orange lines represents different gradient deviations that defines the thresholds.

5.7.4 Implementation in VSIM

To be able to test the constructed algorithm in an online adaption, VSIM is used. The algorithm is run before the simulation starts, generating a reference SOC-trajectory for the chosen driving mission.

Recalculating the reference trajectory

The reference trajectory is never perfect due to the limited input data and the simplified vehicle model being used in the optimisation. If the controller is unable to follow a reference trajectory, a recalculation of the trajectory is desired. Otherwise, the battery level is highly forced by the controller to get to the reference, which causes the vehicle to operate in a way that might be far from optimal. For example if the vehicle is at standstill at a red light where the speed limit is 50 km/h, the vehicle cannot discharge the battery in the way that the reference profile suggest. If there is a high speed segment after the red light, the vehicle is forced to operate all electric to consume the battery energy "saved" when standing still. Recalculating the reference might prevent such behaviour, resulting in a reduced fuel consumption.

The recalculation is triggered when the SOC deviates from the reference SOC trajectory above a threshold. If that happens, the simulation is paused and the optimisation algorithm is run. Only the part of the drive cycle remaining is recalculated. The reference SOC trajectory is updated and VSIM is run again.

6 Results

This chapter presents the results from testing the developed algorithm in Matlab and comparisons of different implemented features, as well as comparison to the DDP/ECMS algorithm developed in [14]. It does also present results from implementing the developed algorithm in VSIM where the two different controllers are compared and validated. The resulting complete algorithm with controller is compared to the CDCS method to evaluate the potential fuel saving.

The test set consist of three recorded drive cycle, which are used to generate results. These are given the names GAC, PFC, and SHC, where their respective recorded speeds, speed limits and if applicable, altitude profiles can be found in Appendix A. The new standard cycle for vehicle classification, WLTP, is also part of the test set. For SHC and WLTP, no information about the speed limits exist. To use these in the segmentation algorithm, an estimation of the speed limits is made, seen in Appendix A. All tests will be done on a laptop with quad core i7 2.8 GHz CPU and Matlab R2015b.

To test different cases of discharge, a few levels are introduced. These are used both in the Matlab and VSIM tests.

- Large
- Medium
- Small
- Standard
- Sustain

Where Large describes close to a full discharge, small a rather low starting SOC of the battery and medium in between large and small. The standard discharge describes a discharge of the battery between a large and a medium discharge. The sustain discharge describes a charge sustaining case, meaning that the starting and final SOC is equal.

6.1 Segmentation algorithm

As can be seen in Table 6.1, the time for the algorithm to run is dependent on the drivecycle duration, as this correlates with how many data points the set of vectors describing the drive cycle contains. From the table one can also notice that the number of segments created and the length of the drivecycle in kilometers have little to none effect on the time to run the algorithm. In Figure 6.1 a plot of how run time depends on the drive cycle time and therefore the number of data points can be seen. The figure shows a linear relation in the time it takes to run the algorithm in relation to the number of data points.

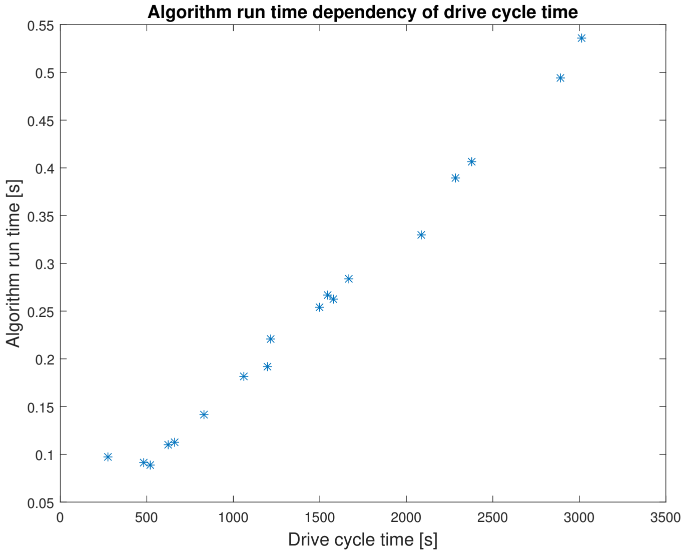


Figure 6.1: *Figure showing how the segmentation algorithm run time depends on the drive cycle length in seconds.*

Table 6.1: *Differences in the run time for the segmentation algorithm shown together with the number of segments created, the distance of the cycle and the drive cycle time for a number of ADASIS recordings that has been run through the segmentation algorithm.*

Time to run algorithm [s]	Number of segments	Distance [km]	Drive cycle time [s]
0.045	6	3.32	274
0.449	30	46.8	2377
0.203	20	22.2	1214
0.347	22	38.9	2087
0.520	30	51.0	3013
0.273	20	30.4	1546
0.268	22	29.6	1576
0.396	24	57.3	2284
0.294	20	32.2	1666
0.086	7	11.5	522
0.085	13	8.23	483
0.202	22	23.7	1197
0.116	7	16.7	661
0.103	8	11.1	622
0.145	15	13.2	832
0.264	19	25.5	1497
0.191	19	16.6	1062
0.522	35	43.7	2890

The results from the segmentation algorithm is seen in Figure 6.2, where the GAC cycle has been run through the segmentation algorithm. It can be seen that the segmentation algorithm misses some parts of the original data. This is mainly due to a minimum segment length required, explained in Section 5.2.

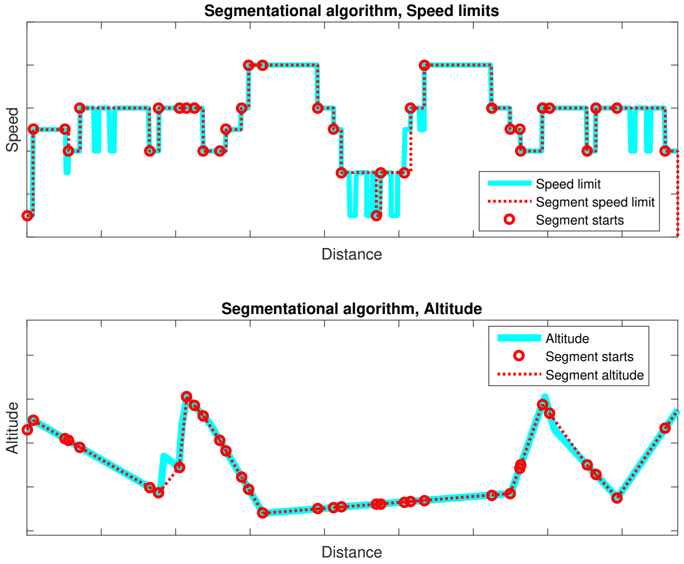


Figure 6.2: *The segmented GAC cycle. The red dots symbolise segment starts, the red dotted line symbolises the data in the segments, and the cyan lines represent the original data, recorded during VCC test drives.*

6.2 Optimisation algorithm

The optimisation algorithm for the energy management is evaluated in Matlab with different drive cycles and settings. A number of features in the algorithm introduced in this thesis is separately tested to find their performance. The result is presented with a plot of the resulting SOC reference trajectory together with a table showing the computational time. In some cases the fuel consumption or number of iterations is also relevant and presented in the table as well.

The original version of the implemented algorithm is defined with the following settings:

- Logically controlled ERAD
- Gear reduction
- Battery model as second order cone
- No acceleration between segments

If nothing else is mentioned about the solver, this version is used. The standard drive cycle used for tests is the cycle named GAC. If nothing else is specified, this cycle has been used.

For each drive cycle, the data to be used in the optimisation algorithms is given the following names:

- Original data, recorded speed *or* DP/ECMS, recorded speed (original data)
- Original data, speed limits
- Segmented, speed limits

Original data means that no segmentation is performed and the original ADASIS data is used with a 1 second resolution. The velocity data can either be from recording the speed of the vehicle or from the speed limits for the same drive. The segmented version is from running the data with speed limits through the segmentation algorithm.

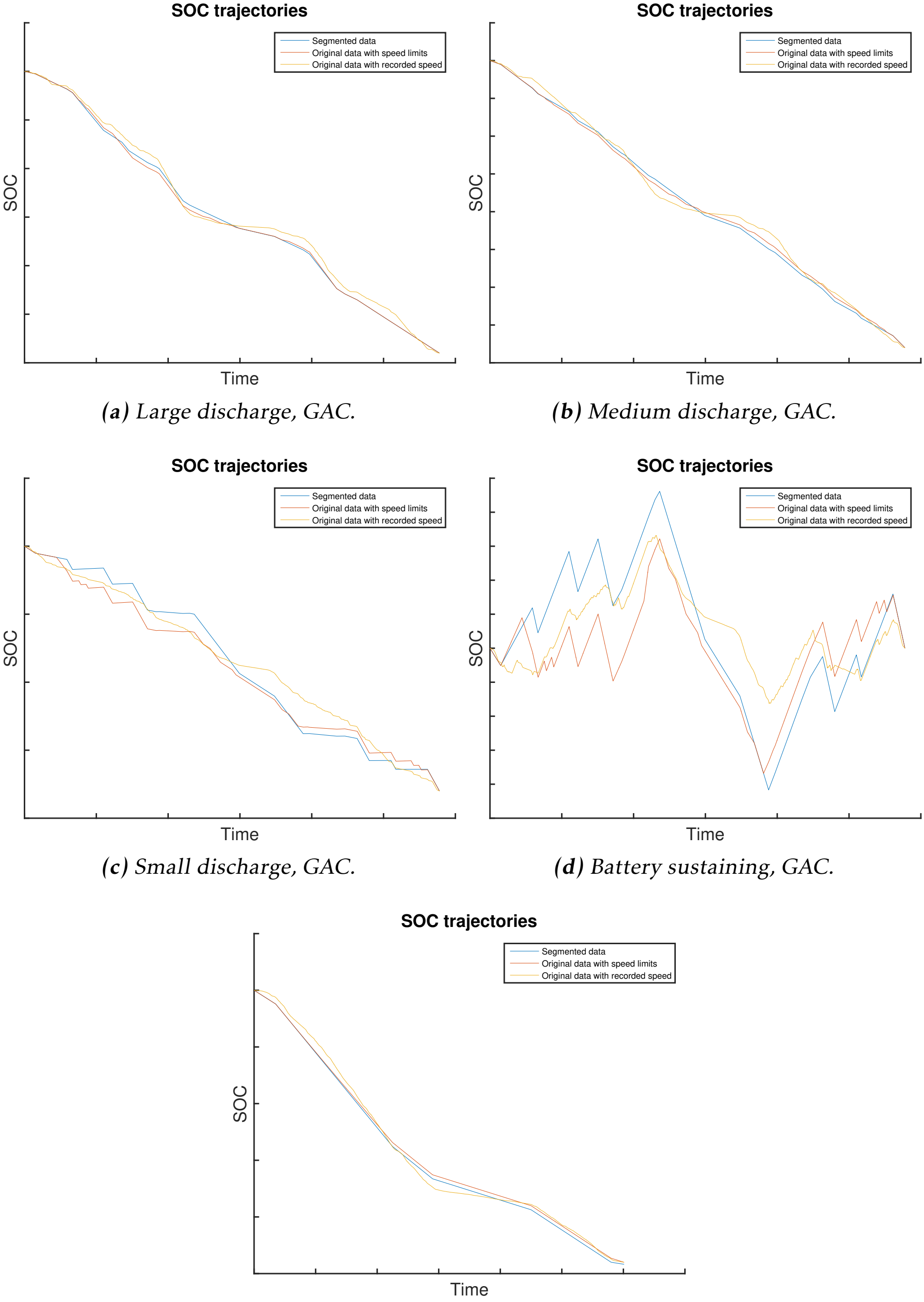
No conclusions can be drawn from the fuel consumption when testing in Matlab with different ways to get the velocity for the same drive cycle since this will generate different versions of the drive cycle. Only when the same drive cycle and velocity profile is used, the fuel consumption from running in Matlab can be compared.

6.2.1 Drive cycle segmentation

This section will show the results from the tests of how the implemented optimisation algorithm performs when used with data that has been run through the segmentation algorithm. The results are compared to when the optimisation algorithm has been used with original data with speed limits and recorded speed. The DDP/ECMS algorithm[14] which the method is also tested and used for normalisation of the presented data.

Table 6.2: *Differences in computational time for the implemented optimisation algorithm with different cycles and different discharges. The column Method describes the input data of different forms. The results from an DDP/ECMS algorithm is also shown in the table for each case and used to normalise from.*

Cycle	Discharge	Method	Computational time
GAC	Large	Segmented, speed limits	4.3
GAC	Large	Original data, speed limits	155
GAC	Large	Original data recorded speed	74
GAC	Large	DDP/ECMS, recorded speed	100
GAC	Medium	Segmented, speed limits	1.2
GAC	Medium	Original data, speed limits	22
GAC	Medium	Original data recorded speed	33
GAC	Medium	DDP/ECMS, recorded speed	100
GAC	Small	Segmented, speed limits	1.7
GAC	Small	Original data, speed limits	17
GAC	Small	Original data recorded speed	22
GAC	Small	DDP/ECMS, recorded speed	100
GAC	Sustain	Segmented, speed limits	1.62
GAC	Sustain	Original data, speed limits	114
GAC	Sustain	Original data recorded speed	18
GAC	Sustain	DDP/ECMS, recorded speed	100
SHC	Standard	Segmented, speed limits	0.81
SHC	Standard	Original data, speed limits	27
SHC	Standard	Original data recorded speed	28
SHC	Standard	DDP/ECMS, recorded speed	100



*(**a**) Large discharge, GAC.*

*(**b**) Medium discharge, GAC.*

*(**c**) Small discharge, GAC.*

*(**d**) Battery sustaining, GAC.*

*(**e**) Standard discharge, SHC.*

***Figure 6.3:** SOC reference trajectories for the methods described in the Table 6.2*

Worth to note from Table 6.2 is the difference in computational speed between the different methods, as well as the difference in the SOC trajectories between the different methods for the same drive cycle, seen in Figure 6.3.

6.2.2 Gear reduction

This section shows the results from the tests of how the implemented optimisation algorithm performs when used with a reduced set of gears, as discussed in Section 5.3.1. The results are compared to tests when the gear reduction is not used. Multiple sets of input data to the optimisation algorithm are used in the tests and only the ones with equal velocity data can be compared in terms of fuel consumption.

Table 6.3: *Differences in fuel consumption, number of iterations and computational time for the implemented optimisation algorithm with and without gear reduction. The results are normalised based on the values from the test with gear reduction and recorded speed from the original data. This table shows the results from a medium battery discharge. The fuel consumption should be compared only between the pairs with equal data speed data.*

Cycle	Method	Fuel consumption per distance	Computational time	Iterations
GAC	No gear reduction, segmented, speed limits	1.10	8.0	17
GAC	Gear reduction, segmented, speed limits	1.14	2.8	7
GAC	No gear reduction, Original data, speed limits	1.02	452	18
GAC	Gear reduction, Original data, speed limits	1.06	66	3
GAC	No gear reduction, recorded speed	0.96	132	5
GAC	Gear reduction, recorded speed	1.00	100	5

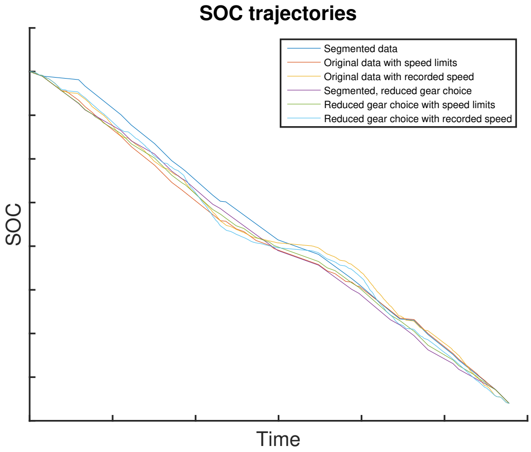


Figure 6.4: *SOC trajectories for the methods described in Table 6.3 where the feature to reduce the number of gears evaluated in DDP in each segment is tested.*

Important to note from Table 6.3 is that the error in fuel consumption from the gear reduction is about the same for all methods, compared to the fuel consumption with no gear reduction. Also worth to note is the difference in computational time between the methods with gear reduction and the methods without gear reduction. The iterations is related to the computational time and can explain the big difference. As can be seen in Figure 6.4, the SOC trajectories for the different methods have minor difference.

6.2.3 Piecewise linear battery model

This section shows results from the tests of how the implemented optimisation algorithm performs when the second order cone is introduced, as discussed in Section 5.5.2, together with the results from when the second order cone is removed and replaced by a piecewise linear model of the battery. Multiple sets of input data to the optimisation algorithm are used in the tests.

Table 6.4: *Differences in fuel consumption, number of iterations and computational time for the implemented optimisation algorithm with and without a second order cone implemented. The input data is in different forms and medium discharge is used. The results is normalised from the values when the second order cone and recorded speed from the original data are used. The fuel consumption should be compared in pairs, the two at the top with each other, the following two with each other etc.*

Cycle	Method	Fuel consumption per distance	Computational time	Iterations
GAC	Cone, segmented, speed limits	1.14	3.8	7
GAC	No cone, segmented, speed limits	1.15	4.3	13
GAC	Cone, original data, speed limits	1.06	67	3
GAC	No cone, original data, speed limits	1.06	375	14
GAC	Cone, recorded speed	1.00	100	5
GAC	No cone, recorded speed	1.00	335	13

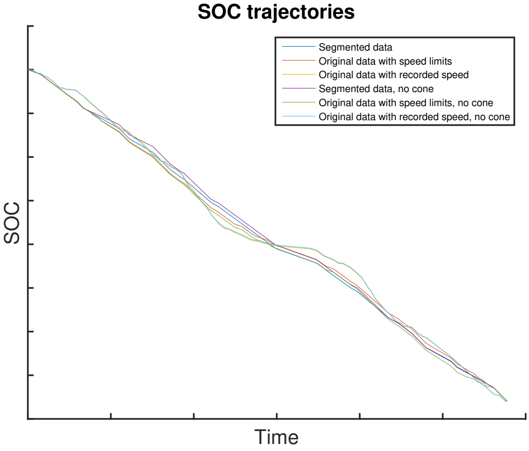


Figure 6.5: *SOC trajectories for the methods described in Table 6.4 where the second order cone is compared to using a piecewise linear battery model.*

Worth to note from Table 6.4 is that the error in fuel consumption between the methods with a second order cone compared to those without a second order cone is quite small. One can note that the method with the implemented cone is faster than the method with the linearised model in all cases. However, the difference in iterations between the methods is quite large, which can explain the difference in computational time. As seen in Figure 6.5, there is some differences in the SOC trajectories for the different methods.

6.2.4 Mean velocity and standard deviation

This section shows the results from the tests of how the implemented optimisation algorithm performs used with segmented data that has additional information about the speed in each segments. The mean speed is used together with standard deviation, where description of the method is seen in Section 5.2.2. The implementation of standard deviation doubles the amount of segments compared to the original segmentation, while the mean speed implementation have the same amount of segments as the original version. The results are compared between using original data with speed limits and using the recorded speed. It is also compared with results from the DDP/ECMS algorithm. The mean speed and standard deviation methods new velocity profiles, meaning two new versions of the drivecycle. Comparing the fuel consumption is therefore not relevant.

Table 6.5: *Differences in number of iterations and computational time for the implemented optimisation algorithm with input data of different forms. The results from an DDP/ECMS algorithm is also shown in the table and used to normalise from. This table shows the results from a medium battery discharge.*

Cycle	Method	Computational time	Iterations
GAC	Segmented, speed limits	1.3	7
GAC	Segmented, mean speed	0.70	4
GAC	Segmented, standard deviation	0.78	3
GAC	Original data, speed limits	24	3
GAC	Original data recorded speed	36	5
GAC	DDP/ECMS solver, recorded speed	100	-

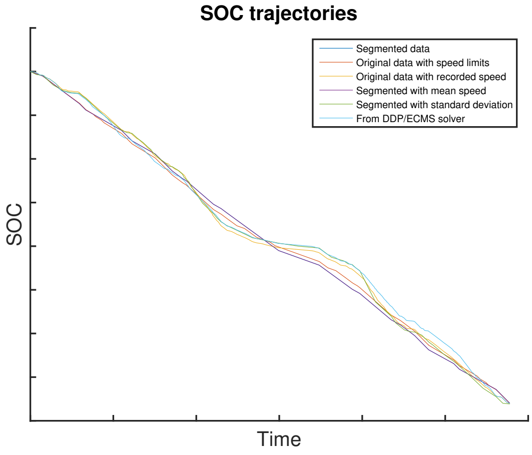


Figure 6.6: *SOC trajectories for the methods described in Table 6.5.*

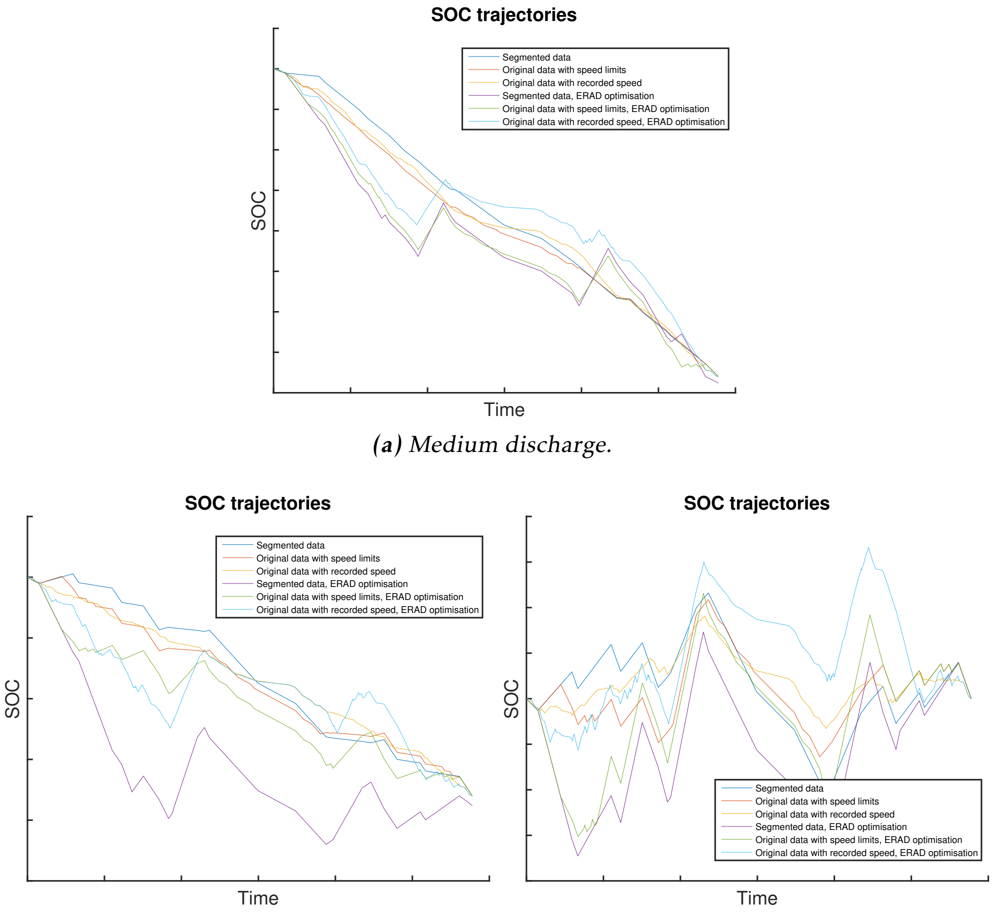
Interesting to note is the difference between the computational time for the segmented methods as can be seen in Table 6.5. The computational time is somewhat higher in the case of segmentation with speed limits than in the cases with mean speed and standard deviation. However, the computational time for each iteration is lower in the cases of mean speed and speed limits than with standard deviation. As can be seen in Figure 6.6, there are some difference between the methods, where one can see that the method with standard deviation follows the DDP/ECMS method quite close.

6.2.5 ERAD optimisation

This section shows the results from tests of how the implemented optimisation algorithm performs when the ERAD is controlled with optimisation, as discussed in Section 2.2.4. To compare and benchmark, the results are compared to using the ERAD controlled with logic. Multiple sets of input data to the optimisation algorithm are used in the tests. The results from ERAD optimisation with the DDP/ECMS algorithm are used to benchmark from. All tests are made without gear reduction to isolate the difference between only changing the control of the ERAD.

Table 6.6: *Differences in fuel consumption and computational time for the implemented optimisation algorithm with ERAD control by logic or optimisation, where the input data are of different forms and for different discharges. Results from an DDP/ECMS algorithm with ERAD optimisation is also shown in the table for each case and used to normalise from.*

Cycle	Discharge	Method	Computational time
GAC	Medium	Segmented, speed limits	0.32
GAC	Medium	Segmented, speed limits, ERAD optim.	1.0
GAC	Medium	Original data, speed limits	18
GAC	Medium	Original data, speed limits, ERAD optim.	73
GAC	Medium	Original data, recorded speed	5.4
GAC	Medium	Original data, recorded speed, ERAD optim.	61
GAC	Medium	DDP/ECMS, recorded speed, ERAD optim.	100
GAC	Small	Segmented, speed limits	0.27
GAC	Small	Segmented, speed limits, ERAD optim.	1.3
GAC	Small	Original data, speed limits	4.3
GAC	Small	Original data, speed limits, ERAD optim.	96
GAC	Small	Original data, recorded speed	6
GAC	Small	Original data, recorded speed, ERAD optim.	90
GAC	Small	DDP/ECMS, recorded speed, ERAD optim.	100
GAC	Sustain	Segmented, speed limits	0.56
GAC	Sustain	Segmented, speed limits, ERAD optim.	1.8
GAC	Sustain	Original data, speed limits	35
GAC	Sustain	Original data, speed limits, ERAD optim.	113
GAC	Sustain	Original data, recorded speed	8.8
GAC	Sustain	Original data, recorded speed, ERAD optim.	139
GAC	Sustain	DDP/ECMS, recorded speed, ERAD optim.	100



(a) Medium discharge.

(b) Small discharge.

(c) Battery sustaining.

Figure 6.7: *SOC trajectories for the methods described in Table 6.6.*

Important to note from Table 6.6 is the difference in many cases when using ERAD optimisation compared to when the ERAD is controlled with logic. The difference in computational time in the "original data" methods versus the "segmented" methods are very large. From Figure 6.7 it can be seen that the SOC trajectories for the different methods differ quite a lot.

6.2.6 Summary

The results from the tests of the optimisation algorithm tells that the computational time can be decreased significantly with the use of segmentation. The computational time can also be reduced by controlling the ERAD with logic and by decreasing the amounts of gears to choose from. The number of iterations differs with the different settings, with more iterations showing that the problem is harder to solve and increasing the computational time. The implementation of mean speed and standard deviation from the mean speed shows some differences in computational time. This is mainly due to fewer iterations needed to converge to an optimum. To replace the second order cone with a piecewise linear model increases the number of iterations needed and the computational time for the algorithm.

In the cases of the replacement of the second order cone with a piecewise linear model and with the implementation of gear reduction, the fuel consumption for the simulations in Matlab are compared. The comparison only holds for the same drive cycles, i.e. the segmented versions can be compared to each other. From the replacement of the second order cone, the difference in fuel consumption for the segmented versions is very small, showing that the implementation of the cone does not effect the solution in a major way. For the implementation of gear reduction, the difference in fuel consumption shows that there might be an effect on the solution. These effects are researched more in the implementation in VSIM as done in Section 6.4.

6.3 Controller

The two controllers described in the method are implemented in VSIM. The parameters in the controllers are tuned and the performance of the algorithm developed in this thesis is evaluated. The vehicle models in VSIM is the most advanced in terms of fuel consumption available, apart from implementation in a real vehicle, which is outside the scope of this thesis. Several cycles and discharge rates are evaluated in order to understand the different scenarios where the method have benefits and drawbacks.

To get the most realistic test method, the simulation is run where the vehicle follows a recorded velocity, while the reference trajectory is calculated based on a segmented profile using speed limits. The controllers are developed with different levels of aggressiveness to test how accurately the reference trajectory should be followed. If nothing else is mentioned, the normalisation of the fuel consumption is made where the original version of the implemented algorithm with a standard tuning of the controller is given the index 1.

The final value of SOC is very important to have in mind. The difference compared to the desired final value is shown in the last column of all tables in this section as *SOC diff %* and calculated as Equation 6.1 where T_{N+1} is the time when the last segment ends.

$$\text{SOC diff \%} = SOC(T_{N+1}) - SOC_{ref}(T_{N+1}) \tag{6.1}$$

A SOC difference at the end of the cycle means that the desired amount of electrical energy hasn't been used. Instead liquid fuel energy has been used, which affects the fuel consumption. A final value above the desired one, a positive difference, indicates that too small amount of battery energy has been used and instead fuel energy, resulting in a higher fuel consumption. To compare the fuel consumption for different control strategies for the same cycle and discharge, the final value of SOC has to be rather similar. By amplifying the reference signal during the last minute of the cycle, the reference trajectory is followed more accurately, resulting in a reduced difference in the final SOC value.

6.3.1 Tuning of the controllers

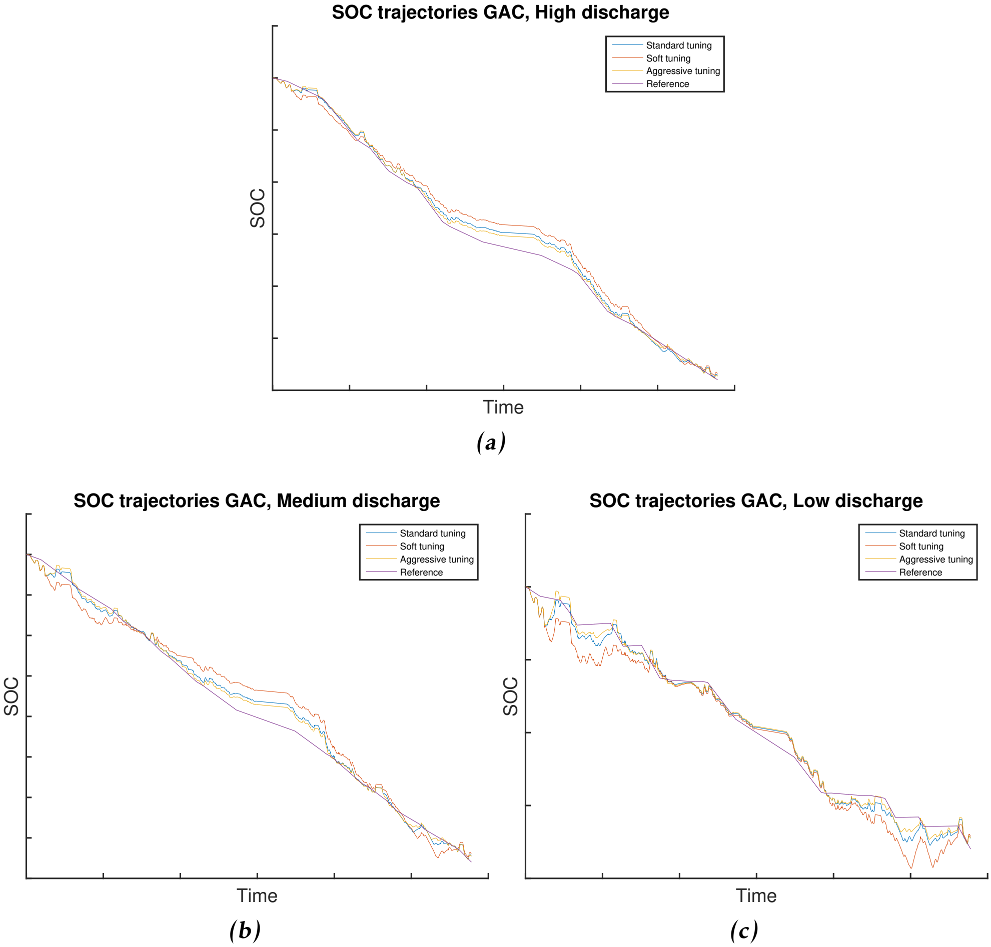
The two controllers are tuned using the GAC cycle to achieve good fuel economy. Three versions of the both controllers are produced. A controller is first tuned to show good results for a number of discharges, called the standard version. This one is then slightly modified to get a more aggressive, as well as a more soft controller. The tuning is made from experience of test drives and studying the power demand in VSIM at different velocities.

Deviation controller

The results from the tuning of the deviation from reference-controller can be seen in Table 6.7. The standard controller performs best or equal to the alternatives for all tested discharges, indicating that the tuning of the standard controller was done in a good way. The SOC trajectories can be seen in Figures 6.8a-c, where a clear difference is seen between the controllers, especially for the small discharge, as seen in Figure 6.8c.

Table 6.7: *Results from simulation with different tuning of the deviation controller and with different discharges using the GAC cycle.*

Cycle	Discharge	Controller tuning	Fuel consumption	SOC diff %
GAC	Large	Standard	1.00	0.9
GAC	Large	Soft	1.01	1.3
GAC	Large	Aggressive	1.00	0.7
GAC	Medium	Standard	1.00	0.8
GAC	Medium	Soft	1.02	1.0
GAC	Medium	Aggressive	1.02	0.7
GAC	Small	Standard	1.00	0.8
GAC	Small	Soft	1.00	0.9
GAC	Small	Aggressive	1.00	0.8



(a)

(b)

(c)

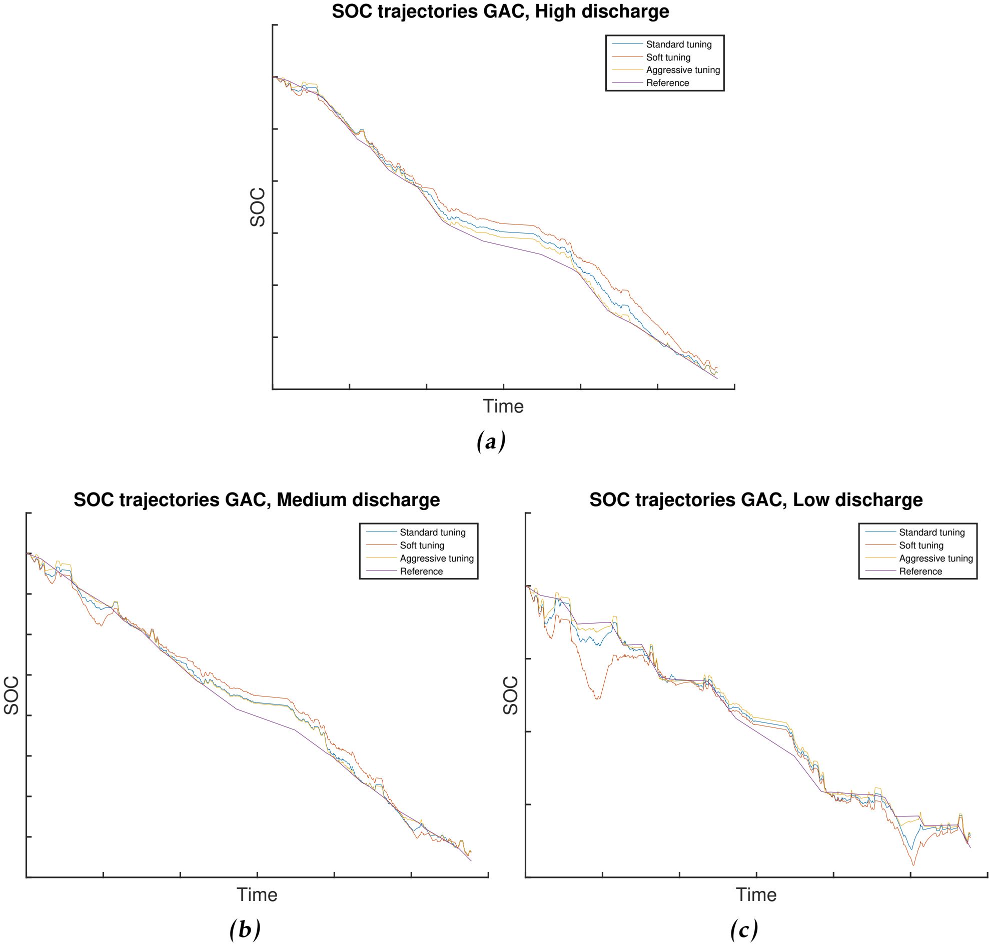
Figure 6.8: *SOC trajectories for the methods described in Table 6.7.*

Gradient controller

The results from the tuned gradient controller is presented in Table 6.8. Here, the softer version of the controller performs slightly better for medium and small discharge. The fuel consumption when using the aggressive controller is equal or higher for all tests compared to the standard controller.

Table 6.8: *Results from simulations with different tuning of the gradient controller and different discharges with the GAC cycle.*

Cycle	Discharge	Controller tuning	Fuel consumption	SOC diff %
GAC	Large	Standard	1.00	1.2
GAC	Large	Soft	1.01	2.1
GAC	Large	Aggressive	1.01	1.1
GAC	Medium	Standard	1.00	1.2
GAC	Medium	Soft	0.99	1.1
GAC	Medium	Aggressive	1.03	1.2
GAC	Small	Standard	1.00	0.9
GAC	Small	Soft	0.99	0.7
GAC	Small	Aggressive	1.03	1.0



(a)

(b)

(c)

Figure 6.9: *SOC trajectories for the methods described in Table 6.8.*

6.3.2 Validation of the controllers

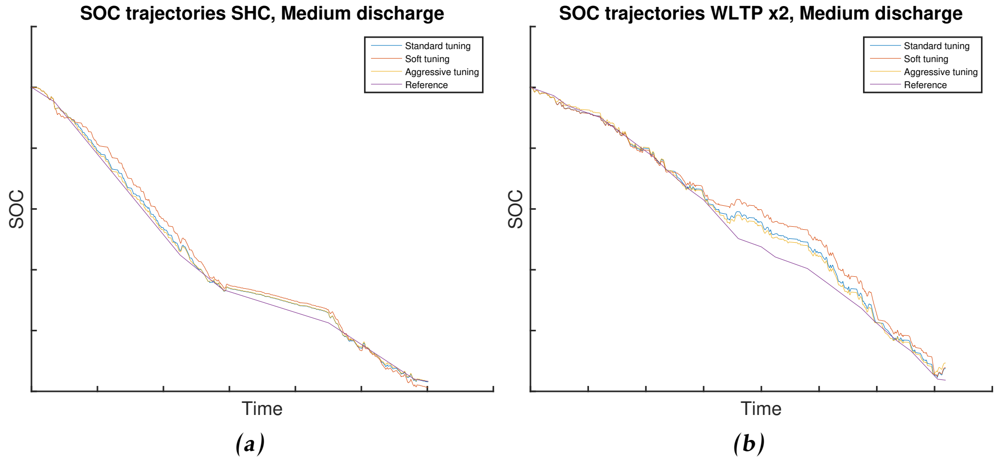
To ensure that the controllers performs well for more than the GAC cycle, a validation with other cycles is presented in this section. Another recorded cycle called SHC and the new standard cycle for fuel consumption, WLTP, is used. To achieve realistic results, the starting SOC of the battery is an important factor. With a large amount of energy available in the battery, the vehicle will run all electric for almost the entire cycle giving minor room to optimise the usage of the ICE. This results in the method in this thesis being very close to CDCS, hence a starting charge that will generate both electric and fuel powered propulsion is chosen.

Deviation controller

As seen in Table 6.9, the only difference in fuel consumption is for the SHC cycle with the soft controller. However the final SOC value for that method is smaller than the result from the other tuning-levels of the controller, implying that too much electric energy was used and fuel saved.

Table 6.9: *Simulation of the different tunings of the deviation controller. Two cycles are tested with the three different tuning levels.*

Cycle	Discharge	Controller tuning	Fuel consumption	SOC diff %
SHC	Medium	Standard	1.00	0
SHC	Medium	Soft	0.99	-1.0
SHC	Medium	Aggressive	1.00	0
WLTP x2	Medium	Standard	1.00	1.9
WLTP x2	Medium	Soft	1.00	2.0
WLTP x2	Medium	Aggressive	1.00	2.8



(a) *(b)*

Figure 6.10: *SOC trajectories for the methods described in Table 6.9.*

Gradient controller

For the gradient controller, the soft tuning does not perform well for the SHC cycle due to the long distance between the target points. The standard and the aggressive tuning gives similar results.

Table 6.10: *Simulation of the tuned gradient controller. Two cycles are tested with the three different tuning levels.*

Cycle	Discharge	Controller tuning	Fuel consumption	SOC diff %
SHC	Medium	Standard	1.00	0
SHC	Medium	Soft	1.12	7.7
SHC	Medium	Aggressive	1.00	0
WLTP x2	Medium	Standard	1.00	1.9
WLTP x2	Medium	Soft	1.00	2.6
WLTP x2	Medium	Aggressive	0.99	1.7

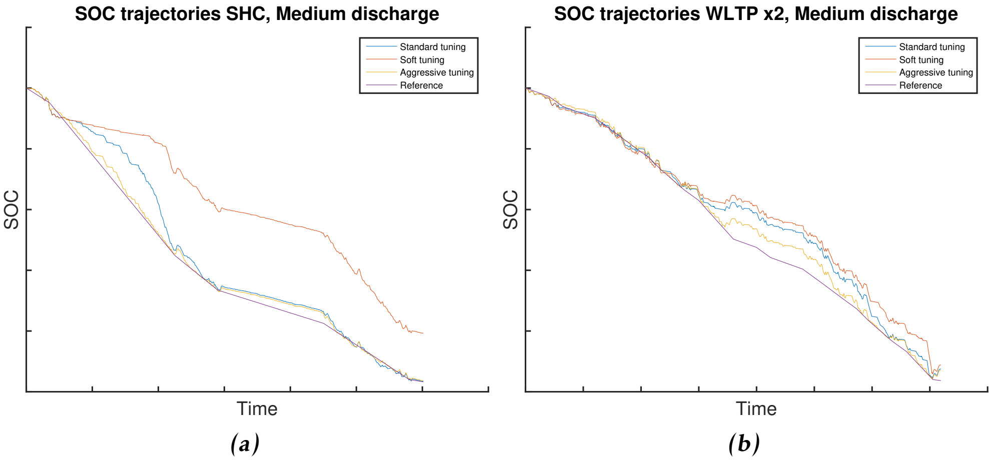


Figure 6.11: *SOC trajectories for the methods described in Table 6.10.*

6.4 Simulation of complete algorithm with controller

To analyse how the different settings of the implemented algorithms affect the fuel consumption, the complete algorithm and controllers are tested in simulation. The computational time for the algorithm is equal to what is presented in Section 6.2, as the exact same algorithm is used in the simulations. The standard tuning of both the deviation and gradient controller are used for all further tests. If nothing else is mentioned, the legend *Standard* in the figures denotes that the standard tuning of the deviation controller and the original settings in the complete algorithm, Section 6.2, are used.

6.4.1 Segmentation compared to recorded speed

A simulation is run where the SOC reference trajectory is calculated based on the recorded speed. This shows how a more accurate reference affects the fuel consumption, compared to using segmented data, see Table 6.11. As seen in Figure 6.12, the reference and actual SOC trajectory are very close during the entire cycle when using recorded speed to generate the reference SOC trajectory.

Table 6.11: *Results from simulation when creating a reference SOC trajectory based on the recorded speed, compared to using segmented data with speed limits.*

Cycle	Discharge	Controller	Fuel consumption	SOC diff %
GAC	Medium	Deviation	1	0.7
GAC	Medium	Deviation, recorded speed	0.98	0

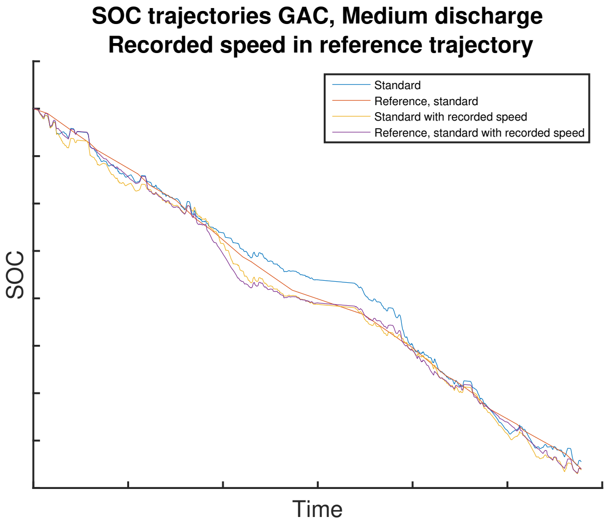


Figure 6.12: *SOC trajectories for the methods described in Table 6.11.*

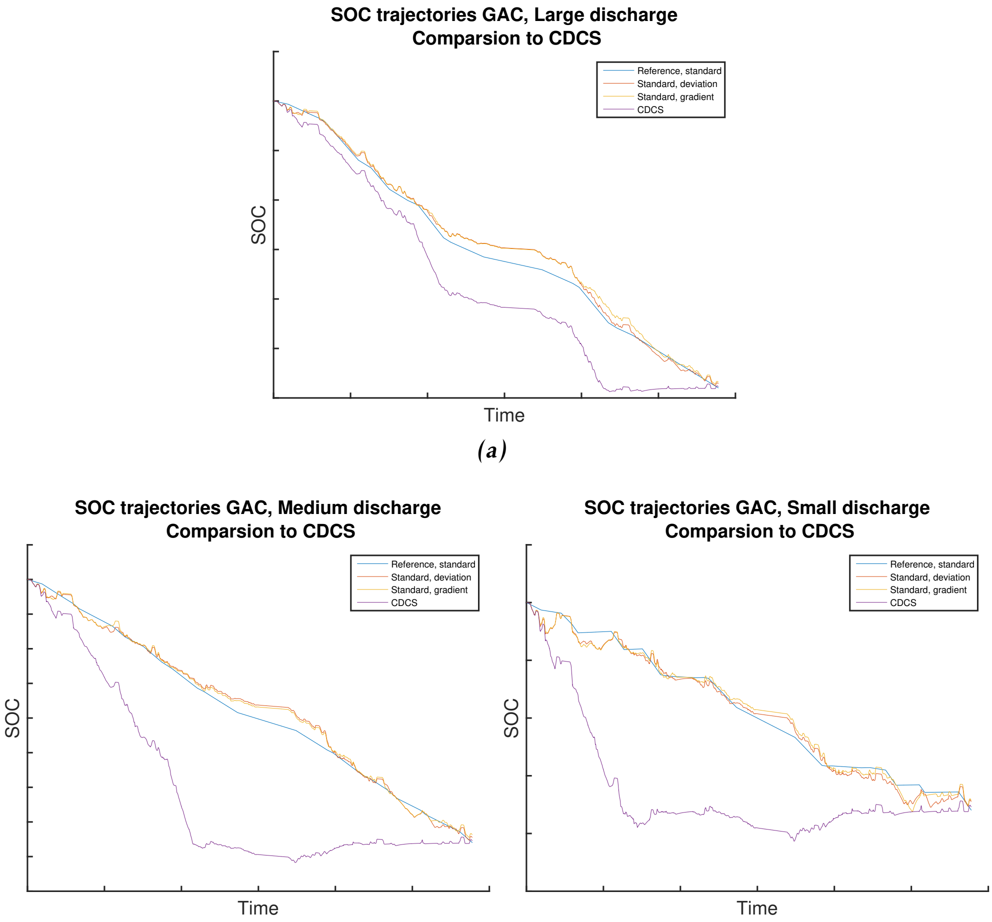
6.4.2 CDCS compared to the implemented algorithm

The algorithm is evaluated against a CDCS-strategy. This is where the potential fuel savings with the method can be seen. The fuel consumption is normalised from the CDCS results.

The GAC cycles is tested with three discharges, with results as Table 6.12. The final value of SOC can be noticed as higher with the controllers than CDCS for all cases.

Table 6.12: *A CDCS strategy simulated for three levels of discharge and compared to simulating with the two controllers and a SOC reference trajectory.*

Cycle	Discharge	Controller	Fuel consumption	SOC diff %
GAC	Large	CDCS	1.00	0.3
GAC	Large	Deviation	0.92	1.3
GAC	Large	Gradient	0.93	1.2
GAC	Medium	CDCS	1.00	0.3
GAC	Medium	Deviation	0.92	1.2
GAC	Medium	Gradient	0.94	1.2
GAC	Small	CDCS	1.00	0.3
GAC	Small	Deviation	0.96	0.8
GAC	Small	Gradient	0.97	0.9



(a)

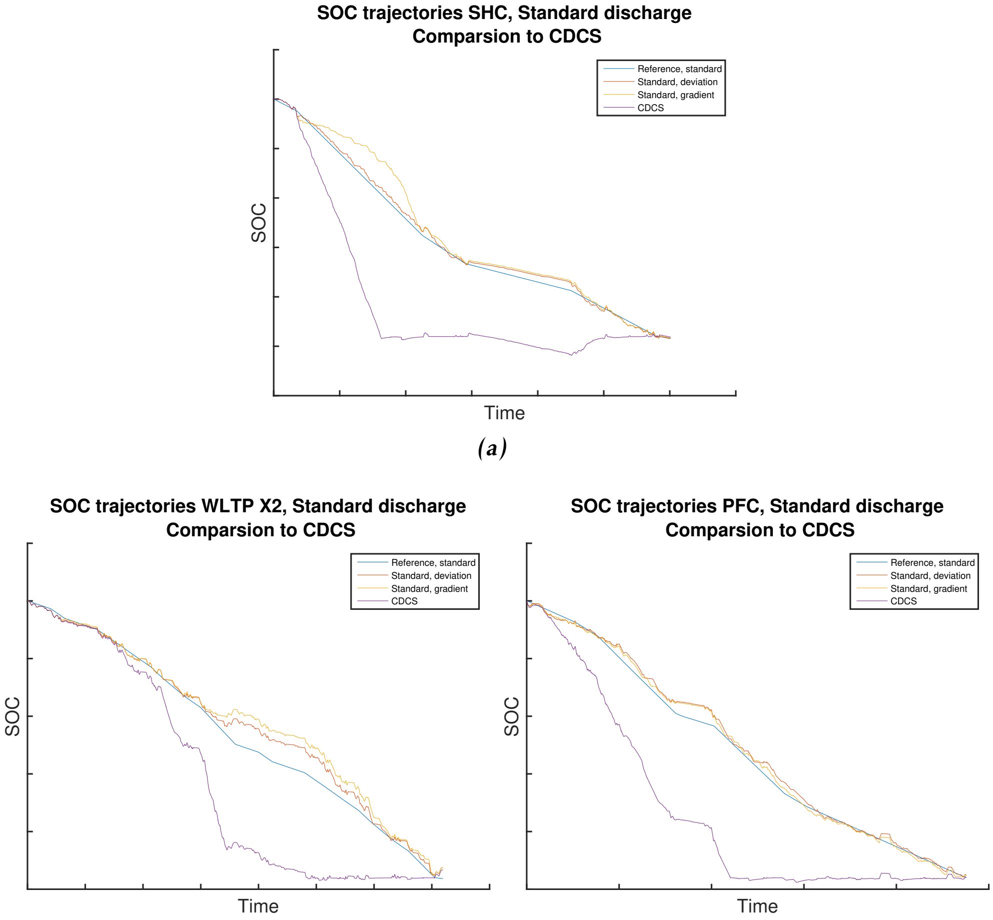
(b) *(c)*

Figure 6.13: *SOC trajectories for the methods described in Table 6.12.*

The results from simulating the SHC, PFC and WLTP cycles is seen in Table 6.13. The larger final SOC diff when running with the controllers compared to CDCS is worth to notice.

Table 6.13: *A CDCS strategy simulated for three drive cycles and compared to simulating with the two controllers and a SOC reference trajectory.*

Cycle	Discharge	Controller	Fuel consumption	SOC diff %
SHC	Standard	CDCS	1.00	0.0
SHC	Standard	Deviation	0.91	0.3
SHC	Standard	Gradient	0.92	0.0
WLTP x2	Standard	CDCS	1.00	1.3
WLTP x2	Standard	Deviation	0.94	1.9
WLTP x2	Standard	Gradient	0.95	1.9
PFC	Standard	CDCS	1.00	-0.0
PFC	Standard	Deviation	0.90	0.4
PFC	Standard	Gradient	0.90	0.3



(a)

(b) *(c)*

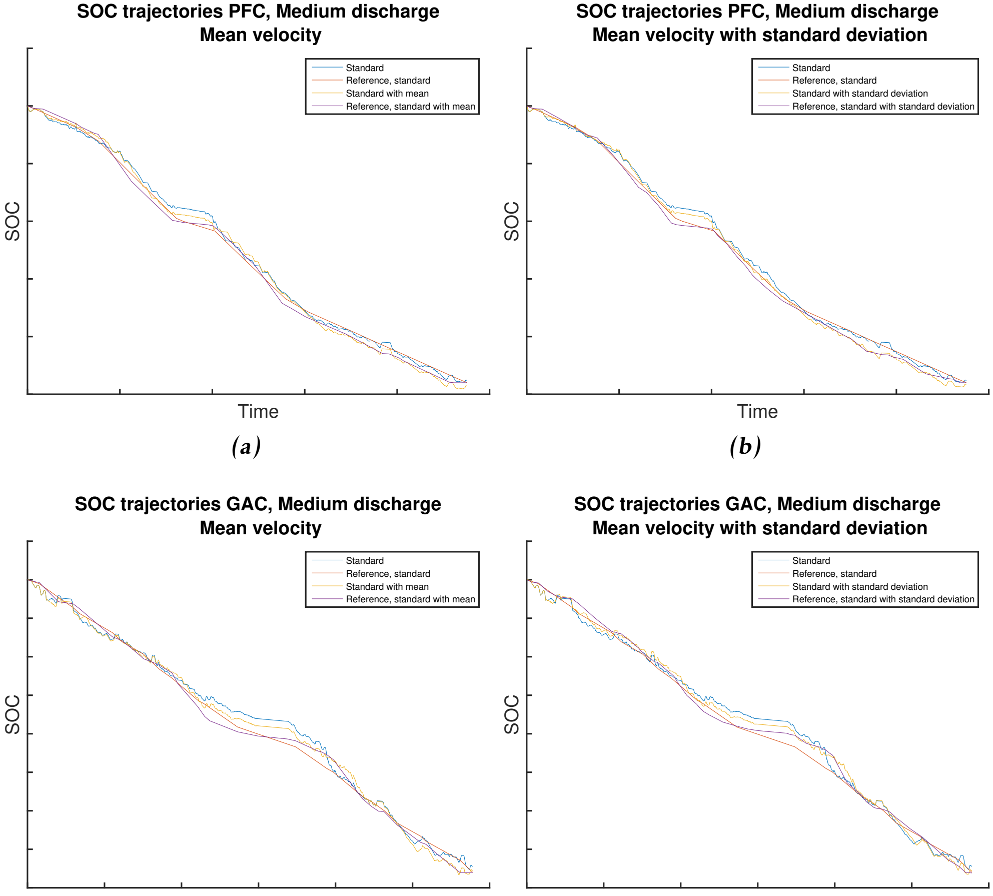
Figure 6.14: *SOC trajectories for the methods described in Table6.13.*

6.4.3 Mean velocity and standard deviation

The usage of mean speed and standard deviation from the mean speed in the segments when calculating the reference SOC trajectory is tested in VSIM, with results in Table 6.14.

Table 6.14: *The results from simulating GAC and PFC when creating the SOC reference profile with mean speed and standard deviation from the mean speed as compared to the original version with segments with the speed as speed limits.*

Cycle	Discharge	Controller	Fuel consumption	SOC diff %
GAC	Standard	Deviation	1.00	0.3
GAC	Standard	Deviation + mean	0.99	0.2
GAC	Standard	Deviation + mean + std	0.99	0.3
PFC	Standard	Deviation	1.00	0
PFC	Standard	Deviation + mean	0.99	-0.5
PFC	Standard	Deviation + mean + std	0.99	-0.3



(a) *(b)*

(c) *(d)*

Figure 6.15: *SOC trajectories for the results from simulating with the setup as described in Table 6.14.*

6.4.4 Recalculating the SOC reference trajectory

The recalculation of the SOC reference trajectory is implemented in VSIM and validated with the GAC cycle. As seen in Table 6.15, the fuel consumption is lowered when recalculating the SOC trajectory with similar SOC difference. Figure 6.16 shows the step in reference at the middle of the drive cycle when the recalculation is made.

Table 6.15: The resulting fuel consumption and SOC difference testing recalculation of the SOC reference trajectory as compared to the original version without recalculation of the SOC trajectory.

Cycle	Discharge	Controller	Fuel consumption	SOC diff %
GAC	Medium	Deviation	1	0.7
GAC	Medium	Deviation with recalculation of the SOC reference	0.995	0.8

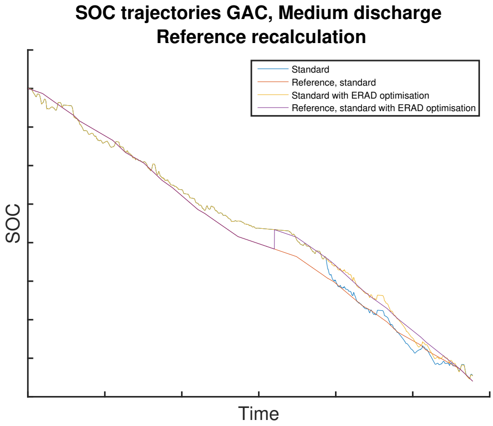


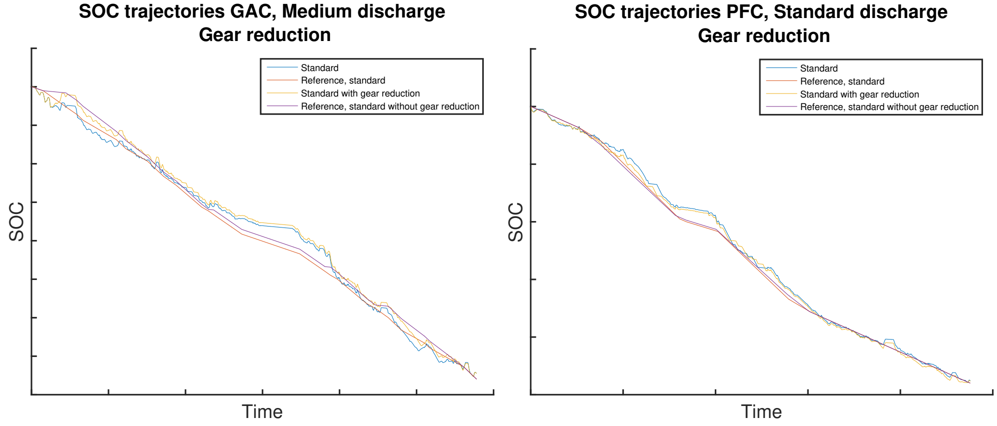
Figure 6.16: SOC trajectories for the method described in Table 6.15.

6.4.5 Gear reduction

When removing the reduction of gears, no noticeable difference in the fuel consumption can be seen, see Table 6.16. Interesting to notice is the difference in SOC reference trajectory, Figure 6.17. Due to allowing the highest gear at a low velocity when not using gear reduction, the trajectory differs.

Table 6.16: *The resulting fuel consumption and SOC difference when simulating with a SOC reference profile calculated without the reduction of gears as compared to the original version with gear reduction.*

Cycle	Discharge	Controller	Fuel consumption	SOC diff %
GAC	Medium	Deviation	1.00	0.8
GAC	Medium	Deviation without gear red	1.00	0.8
PFC	Medium	Deviation	1.00	0.0
PFC	Medium	Deviation without gear red	1.00	0.3



(a) *(b)*

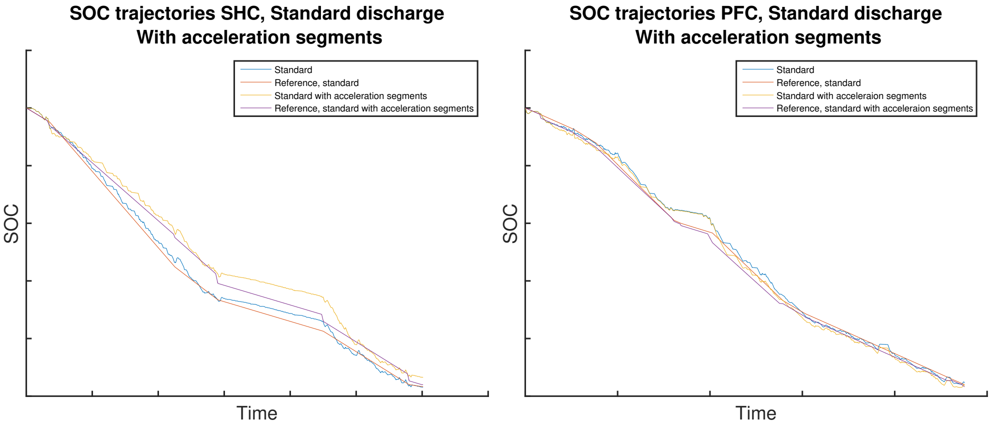
Figure 6.17: *SOC trajectories for the methods described in Table 6.16.*

6.4.6 Acceleration segments

When adding acceleration segments in the calculation of the SOC reference trajectory, as described in Section 5.3.2, a noticeable difference in reference is seen for SHC and a minor difference is seen for PFC, Figure 6.18. The resulting fuel consumption and SOC difference is presented in Table 6.17.

Table 6.17: *The resulting fuel consumption and SOC difference from simulating the SHC cycle with acceleration segments as compared to the original method without acceleration segments.*

Cycle	Discharge	Controller	Fuel consumption	SOC diff %
SHC	Medium	Deviation	1.00	0.7
SHC	Medium	Deviation w/ acceleration	1.02	1.3
PFC	Standard	Deviation	1.00	0.3
PFC	Standard	Deviation w/ acceleration	0.99	0.2



(a) *(b)*

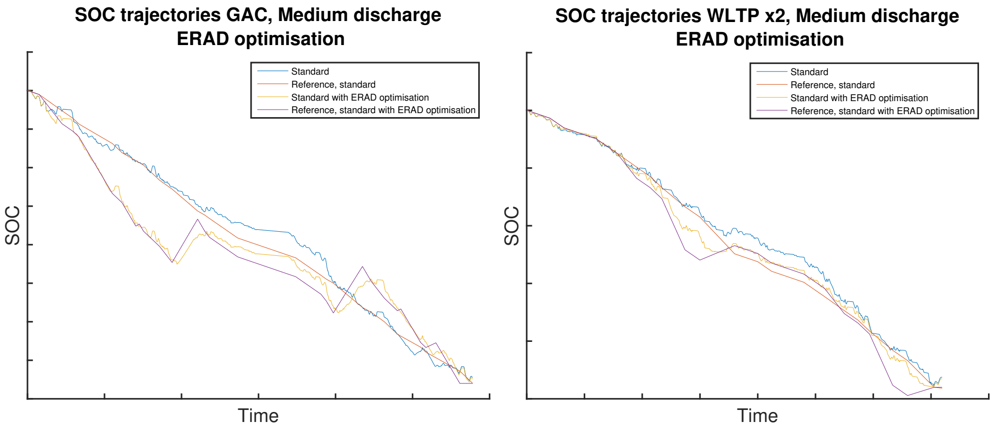
Figure 6.18: *SOC trajectories for the methods described in Table 6.17*

6.4.7 ERAD optimisation

When the ERAD is controlled through optimisation instead of with logic, the resulting fuel consumption is slightly higher for GAC and lower for the two WLTP cycles as can be seen in Table 6.18. As seen in Figure 6.19, there is a major difference between the SOC reference trajectories, while the fuel consumption is rather similar.

Table 6.18: *The resulting fuel consumption and SOC difference when simulating with a SOC reference profile calculated with optimisation of the ERAD usage as compared to the original version with the ERAD controlled by logic.*

Cycle	Discharge	Controller	Fuel consumption	SOC diff %
GAC	Medium	Deviation	1.00	0.8
GAC	Medium	Deviation ERAD optim	1.01	0.6
WLTP x2	Standard	Deviation	1.00	1.9
WLTP x2	Standard	Deviation ERAD optim	0.98	1.5



(a) *(b)*

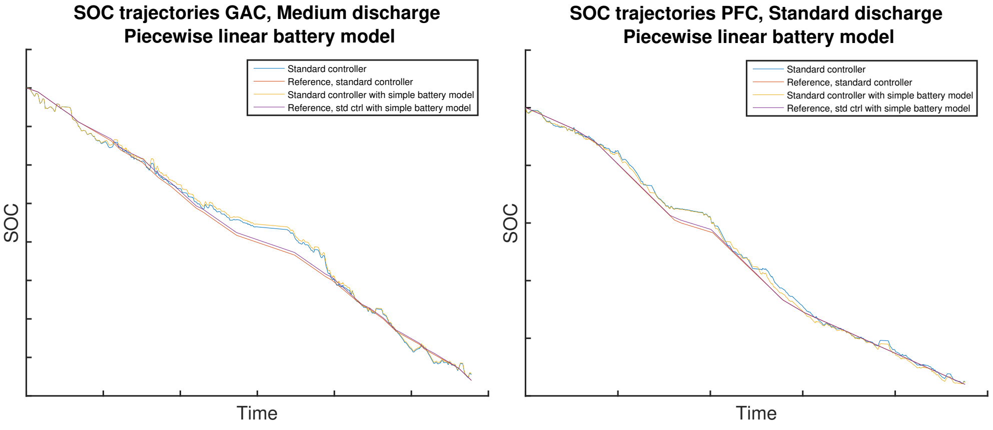
Figure 6.19: *SOC trajectories for the method described in Table 6.18*

6.4.8 Piecewise linear battery model

The second order cone is removed from the battery model and replaced by a piecewise linear battery model. No change in fuel consumption is seen, Table 6.19, however a minor difference in SOC reference trajectory for both of the cycles can be seen, see Figure 6.20.

Table 6.19: *The resulting fuel consumption and SOC difference from simulating the GAC cycle with the battery modelled with piecewise linear functions compared to the original version with a second order cone.*

Cycle	Discharge	Controller	Fuel consumption	SOC diff %
GAC	Medium	Deviation	1.00	0.7
GAC	Medium	Deviation w/ removed cone	1.00	0.8
PFC	Medium	Deviation	1.00	0.0
PFC	Medium	Deviation w/ removed cone	1.00	0.3



(a)

(b)

Figure 6.20: *SOC trajectories for the methods described in Table 6.19.*

6.4.9 Summary

The complete algorithm together with the controller tested in VSIM show a significant possibility to save fuel compared to a CDCS strategy for all tested cycles. The standard tuning of the controllers gives the overall best performance overall compared to a softer or more aggressive tuning. All of the settings introduced in the method, Chapter 5, are evaluated individually in order to create a good basis for the discussion. In general the results from the tests can be summarised as the original version of the implemented algorithm, presented in Section 6.2, gives promising and reliable results in terms of fuel savings. The recalculation of the SOC reference trajectory is implemented with good results.

7 Discussion

The results shows the great potential for the implemented algorithm to save fuel for the tested plug-in hybrid. Comprehensive results are produced and presented in Chapter 6 to show how the developed algorithm perform. Tests of the different features that reduces the computational time while increasing the accuracy of the results are shown. To get a reliable results, a number of different drive cycles was used for testing and validation.

7.1 Optimisation algorithm

The implemented algorithm is a global optimum strategy for the approximated problem presented in the thesis. Since the method is built for the approximated problem, it should be benchmarked against a global optimum strategy such as DP, with a more advanced problem setup. To really examine the method, it should also be benchmarked against a real time strategy. Those two tests would better help to understand the performance of the implemented algorithm.

7.2 Simulation of complete algorithm

When simulating a number of drive cycles, a reduction in fuel consumption of 5-10% compared to CDCS is achieved. Only a minor difference is seen in fuel consumption when simulating using different ways to calculate the SOC reference trajectory, as discussed more detailed in this chapter. The difficulty behind designing the EMS is based on several things, including a simple design of the controller, simple vehicle models in the algorithms and difference between speed limits and the actual driven velocity. The favourable results can be seen as a good trade-off between these factors, where an additional significant reduction in fuel consumption requires an overall improvement while keeping the computational time low.

7.3 Data accuracy

The altitude data from ADASIS used in the segmentation is often scant, meaning very infrequent updates of the altitude compared to the actual inclination. To correct this and approximate an altitude profile for the drive cycles, the data has been interpolated. This can lead to errors and does most likely smooth out the inclinations.

The speed limits provided from ADASIS will in most cases not be a correct approximation of the actual driven velocity. There are many factors that will affect the speed of the vehicle, most important the driver and the traffic. However, even with the bad approximation of the speed that is the speed limits, the results produced are quite good. This is discussed more in Section 7.6.4.

7.4 Vehicle model

As the vehicle model used in the implemented algorithms is without dynamics and with piecewise linear component models, which in themselves are approximations, the accuracy they provide can be set to question. As the offline simulations are run only using the implemented algorithm with these simple vehicle models, the fuel consumption in those cases should only be used to compare results with the same drive cycle. This gives an indication of how the different implementations of the algorithm performs compared to each other.

To get a better approximation of the fuel consumption and how it performs compared to other methods one should look at the VSIM implementation. In VSIM, the vehicle models are much more advanced, thus suitable for generating results for how the implemented algorithm is affecting the fuel consumption. The results can also be compared with how a real vehicle would behave with different EMS algorithms.

The need for a more complex vehicle model in the algorithm is connected with the low accuracy of the ADASIS data. A better vehicle model will most likely give a result closer how the vehicle is operated in a reality. However, since the data from ADASIS is not accurate and a more complex model with added dynamics would increase the computational time for the algorithm, increasing the complexity is not regarded necessary.

The method of optimising with quite simple models without dynamics and with input data on a segmented form, as done in the implemented algorithm performs well in the test sets used. It is interesting to see that despite the simplifications done, the implemented algorithm can produce a SOC trajectory with high quality.

7.5 Connecting DDP/ECMS and ECOS

The method to use DDP/ECMS for choice of operating mode and ECOS to find the correct equivalence factor was found to be quite difficult in terms of connecting the algorithms. The problem formulation in the given DDP/ECMS-algorithm is quite advanced and was difficult to translate for the convex optimisation problem formulation. Due to this, a minor difference between the solvers might sometimes exist, which is handled in the control law when updating λ.

The authors of [7], suggested a simple method of updating λ between each iteration of the algorithm. This method was found to require improvement for our implementation, especially when using optimisation of the ERAD state. By analysing the results, the algorithm is improved with several new ways of updating λ for a faster and robust algorithm for all possible optimisation settings. The drawback with the DDP/ECMS method is that a very precise λ is sometimes required to get the exact correct start and end value of the SOC trajectory. The way of decreasing the weight factor q and stopping when below a threshold gives good performance. To continue iterating with a very small q gives no noticeable difference in the SOC reference trajectory from the convex optimisation.

The first guess of λ is of great importance, especially for the computational time. With reduced set of gears, logically controlled ERAD and segmented data, the problem is most often solved with only a few iterations of the implemented algorithm. For this problem setup, the sensitivity to a guess far from the correct one is low and the problem is always possible to solve with a low number of iterations of the algorithm.

For the optimisation based control of the ERAD, the sensitivity to the initial λ is rather high, thus the update of λ needs to be less rapid. For example if the λ is slightly too low, ECOS might produce a value substantially higher the correct, which the update law needs to handle. The method to always iterate towards a reduced RMS-value of λ is one of the keys behind solving problems where the sensitivity to correct λ is high. By lowering the coefficient q that controls the update of λ when the RMS-value is not decreasing, the problem is possible to solve with good results.

7.6 Detailed discussion of results

In this section the results presented in Chapter 6 as well as the different methods used in the optimisation to produce the results is analysed and discussed.

7.6.1 Test set

The tests have been made using the entire, or parts of the test set. The test set contains four different drive cycles, each with different characteristics. A set of only four different drive cycles is however quite small and for three of these only a standard discharge is used. To make an detailed conclusions about the implemented algorithm and how it effects the vehicle fuel consumption, the test set would have to be increased. The set of four drivecycles used to provide the results in Chapter 6 is however enough for a report and to give a conclusion of the performance of the algorithm.

7.6.2 Drive cycle segmentation

The results from when the segmentation algorithm is used as input to the optimisation algorithm is seen in Section 6.2.1. The resulting difference in computational time between the methods is very large, as seen in Table 6.2. The computational time of the algorithm with segmented data as input only need 0.4-4.5% of the time it takes for the DDP/ECMS algorithm with the original data with real speed as input.

The fuel consumption in the Matlab simulations is however as discussed above, very dependent on the implemented models, it is also not comparable due to the use of different velocities in the drive cycles. Therefore the algorithm was implemented in VSIM and used to generate a reference SOC trajectory to follow. With this implementation, the results can be seen in Section 6.4.1. From those results, one are able to see that with the more complex models, the resulting fuel consumption between the different methods is very small. This shows the true strength of the implemented segmentation algorithm. A very large decrease in computational time is observed, while the fuel consumption is about the same for the methods. That the results from the implementation in VSIM could be so good, even with the simplifications made could probably be derived from the fact that the segmentation still does provide a good SOC trajectory. The controller is designed to follow the SOC trajectory while still taking the vehicle characteristics into account.

7.6.3 Controller

The design of the two controllers are simple and the parameters are tuned mainly based on the power request for the GAC cycle and testing. Tuning the parameters in a more advanced way and based on more cycles might reduce the fuel consumption even more. However what can be seen is that a rather quick tuning of the controller gives a major improvement in fuel economy compared to CDCS.

In the GAC cycle, the vehicle is stopped in the middle of the cycle, while the trajectory is based on a speed limit of 50km/h. The SOC can be seen to differ increasingly from the reference trajectory during stand still, causing the controller to change to a more aggressive behaviour. This can be solved with recalculation of the SOC reference trajectory, as seen in Section 6.4.4. This results in a further 0.5% decrease in fuel consumption compared to using the original reference trajectory.

7.6.4 Mean velocity and standard deviation

The data about the mean speed and standard deviation is not available from ADASIS. These factors are however something that might be able to be estimated with enough measurements from the same road and same driver over time. With enough measurements, it could be a much better approximation of the velocity profile than speed limits, especially in cities due to the traffic. For the purpose of studying the benefit of using this information, the mean speed and standard deviation is calculated based on the recorded speed.

As can be seen from the results in Section 6.2.4 the computational time per iteration is lower for the test with mean speed and speed limits than in the test with standard deviation. The implementation using standard deviation doubles the amount of segments, which explains the increased time per iteration. This does also lead to around twice as much memory needed to run the algorithm, compared to using speed limits or mean speed.

As before, the fuel consumption for the offline runs cannot be trusted, therefore the results from the VSIM implementation that can be seen in Section 6.4.3 is studied. The results shows about 1% decrease in fuel consumption for both cases and a clear difference in the SOC reference profile compared to using speed limits. This tells us that if the data is available, both solutions could improve the fuel economy. In particular using mean speed over the segment show promising results, the computational power required is not increased while the fuel consumption is lowered.

7.6.5 Acceleration segments

When testing with acceleration segments, the two cycles evaluated give opposite results. For GAC, the reference SOC trajectory differs significantly from the one without acceleration and results in a 2% higher fuel consumption. The PFC cycle on the other hand gives a 1% decrease in fuel consumption. The method to place acceleration segments where the speed limits changes can be questioned, both since the size of the problem increases with more segments and the issue of finding the correct places where the driver accelerates. From the velocity profiles seen in Appendix A, it can be noticed that a change in speed limit doesn't necessary mean that the driver changes the speed of the vehicle where the sign is.

7.6.6 ERAD optimisation

When changing to generate a SOC reference trajectory when controlling the ERAD through optimisation, a significant difference is seen in computational time and SOC reference trajectory compared to a logically based control, see Section 6.2.5. The computational time is more than tripled when using optimisation of the ERAD. Results from testing this method in VSIM, presented in Section 6.4.7, show surprising results. The expected result was that the more advanced optimisation would yield a reference trajectory that lowered the fuel consumption compared to a logically based trajectory. This was seen for the double WLTP cycle where a 2% decrease in fuel consumption is presented, however, for the GAC, the fuel consumption increases by 1%.

That the controllers are tuned based on a profile generated by ERAD logic and the large difference between speed limits and recorded speed for GAC could be an explanation. The simple design of the controller might also be the factor for the surprising results. The components controlled is the ICE and ISG and not the ERAD, therefore the benefits of creating a reference trajectory with ERAD optimisation might even be lost.

7.6.7 Gear reduction

The implemented gear reduction is done so that the algorithm does not have as many gears to choose between in each segment, thus reducing the problem. Improvement has been shown in the behaviour of the gear choices, some of which were quite unrealistic before the implementation of the gear reduction. The highest gear could be chosen at a quite low speed as it theoretically can be the most effective way to propel the vehicle, this is however a situation that would not happen during normal driving. The results as shown in Section 6.2.2 shows that the reduction of computational time between the different cases is substantial. The great benefit from using gear reduction is the reduced number of iterations required, seen in Table 6.3. From the results from the implementation in VSIM as seen in Section 6.4.5, it can be seen that the fuel consumption is the same in the case with gear reduction as the case without. This means that the reduction of computational speed has been achieved without a loss of accuracy for the resulting fuel consumption.

7.6.8 Piecewise linear battery model

The implemented second order cone makes it possible to use a more advanced battery model. Without the cone, the battery is modelled as piecewise linear and with the cone the model is a second order equation with more variables used in the model. The results from the offline simulation shown in Section 6.2.3, shows that the piecewise linear version is slightly slower in computational time, but faster in computational time per iteration than the version with a second order cone. A better initial λ might therefore have shown a different result in the computational time. The results from the VSIM implementation shown in Section 6.4.8 shows that the fuel consumption is the same in both cases. Therefore, the use of a more complex cone might not be necessary. A case where a cone might be beneficial is when SOC varies much during the same cycle. This is due to that the voltage-dependency of SOC is included in the cone but not in the piecewise linear model.

7.7 Reduction of emissions

An interesting aspect of combustion engine vehicles is emissions. In the implemented algorithm, the only optimisation variable is the fuel consumption. Emissions of CO2 are highly connected to the fuel consumption and will be reduced when using this method, as compared to CDCS. Another important emission is NOx, which depend on e.g. the temperature of the catalytic converter. As the implemented strategy leads to a blended behaviour in how the vehicle is propelled, the catalytic converter will cool down when propelling the vehicle using only electrical energy, before the combustion engine is started anew. This could lead to increased NOx emissions, however the benefit with this strategy is that in urban areas, electrical energy is in most cases used for propelling the vehicle. NOx emissions is most harmful in cities and the method might reduce problems with high emission there.

8 Conclusions and future work

This chapter presents the conclusions made in the thesis. It reflects the objective of the thesis in Section 1.4 and make conclusions to the questions asked therein, based on the results. The chapter does also present the future work to what is recommended for a continuation of the work presented in the thesis.

8.1 Conclusions

The conclusion for different aspects of the thesis is presented in this section.

- **Optimisation method.** Convex optimisation is successfully introduced to solve the problem and to replace the root finding algorithm. The implemented algorithm is efficient and does provide a SOC reference trajectory with similar characteristics as the DDP/ECMS algorithm[14], which the implemented algorithm is based on.
- **Computational power.** The developed optimisation-based EMS is very efficient, as confirmed by simulations in Matlab/VSIM. The computational time is greatly reduced compared to the DDP/ECMS algorithm, in many cases decreased to about one hundredth of the time. With segmentation of the input data, the problem to solve is reduced, correlated to a limited memory needed for running the algorithm.
- **Fuel consumption.** Implemented in VSIM with more advanced models than in the optimisation algorithm, the developed EMS shows very promising results:
    - The fuel consumption compared with a CDCS method can be reduced by as much as 10%.

  - The difference showed in the Matlab simulations when using no gear reduction has small or no effect in the VSIM simulations.
  - There are no substantial fuel savings seen when controlling the ERAD with optimisation. This is most likely due to the controller, more tests should be done with the suggested improvements in Section8.2.
  - When using recorded speed to generate the SOC reference trajectory, the fuel consumption in VSIM is similar to when using segmented data with speed limits.

  From this in can be concluded that the standard settings used in the implemented algorithm, as presented in Chapter 6 provides good results in terms of fuel consumption. The savings when using other settings is minor.

- **Operating modes.** The generation of the SOC reference trajectory using logic to control the ERAD and gear reduction is tested with promising results. The reduction of operating modes and complexity of the problem to solve, while keeping good results is very desirable. This combination shows great promise for an eventual implementation in a vehicle with limited computational power and memory, since the EMS then need to be as fast and efficient as possible. To control the ERAD with logic in the algorithm does in many cases reduce the computational time by more than 50% compared to controlling through optimisation in the segmented versions. When using gear reduction, the computational time is in this case also reduced by more than 50%.

- **Battery model.** The effects of implementing the battery model as a second order cone instead of using a piecewise linear battery model seems to be negligible. The difference in the fuel consumption and computational time between the cases is small. Therefore, the implementation of the second order cone instead of that of a piecewise linear model can be unnecessarily complicated.

- **Initial λ.** Simulations shows that the computational time is very dependent on the initial λ. With a bad guess, many iterations can be needed for convergence, where a good guess could give convergence to an optimum in only a few iterations.

- **Variables in the input data.** By replacing the speed limits with mean speed and standard deviation, a better estimation of the velocity profile is given. A reduction of 1% in the fuel consumption is achieved, however the mean speed and standard deviation is difficult to estimate in advance. In relation to the reduction of fuel consumption made from implementing the algorithm compared to the CDCS strategy, the fuel savings with this additional data are marginal. From this the conclusion can be drawn that data about speed limits is sufficient.

8.2 Future work

This section presents our suggestions to continue with the work in this thesis.

- A more advanced controller should be constructed and implemented into VSIM to evaluate how different SOC reference trajectories affect the fuel consumption. The one constructed performs quite good for the test set, but there is room for improvement. An issue is that when the SOC is close to the trajectory, the engine on/off thresholds are only based on the speed of the vehicle which gives an oscillatory behaviour when trying to follow the reference. A more advanced method could take this into account and also change the control parameters based on the gradient of the current part of the trajectory.
- To be able to better understand the results, more tests can be run. Especially interesting would be to record more cycles using ADASIS, especially with more altitude data to be able to better test the algorithm.
- To improve the computational time for the algorithm, a way to guess the initial λ should be constructed. A suggestion could be that it could use drive cycle data and desired discharge of the battery, and from those variables make a guess about what λ value should be used initially in the algorithm.
- As of now, the algorithm is only used and tested in Matlab. The results of compiling the code to C and to run tests would be very interesting. An implementation like this would most likely speed up the algorithm significantly. If this was done, it would then be interesting to see the results from a vehicle implementation. To be able to test the algorithm in a vehicle environment would help to better understand the results.
- Implementation of more advanced models could be done, such as the cost for starting the ICE. With the current use of a low number of segments, the number of engine on/off decisions are limited. This means that the costs most likely will have a very minor influence over the whole cycle.
- The development of the EMS has been done with regards only to the fuel consumption. Aspects such as driveability and emissions would be very interesting to analyse.
- The results from comparing the implemented algorithm with a global optimum strategy as pure DP with a more advanced problem setup would be very interesting to evaluate the performance.
- There is a large difference when creating SOC reference trajectories using a logically controlled ERAD compared to an ERAD controlled by optimisation. By improving the logic, that difference could be reduced, however the best results should be produced when using optimisation. Therefore, to continue and develop the solver for ERAD optimisation would probably be the best way forward in order to achieve the best reference trajectories to save fuel.

Appendix

A
Drive Cycles

A.1 SHC

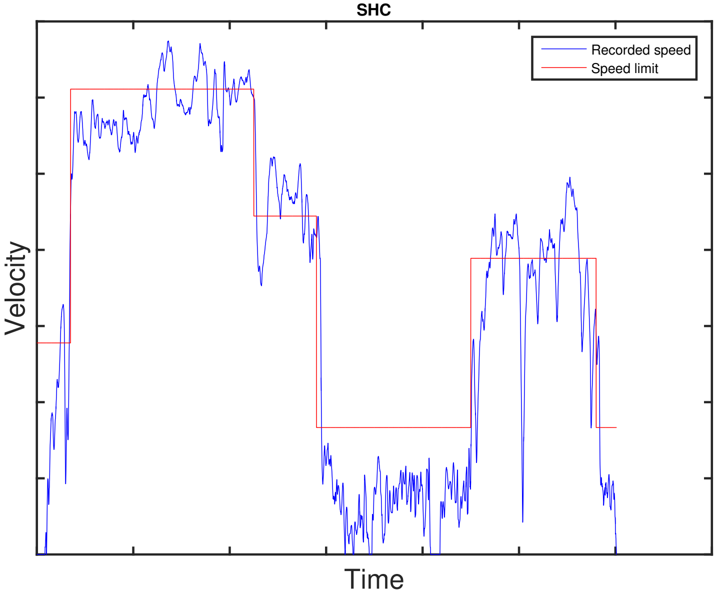


Figure A.1: *The speed of the SHC cycle and estimated speed limits.*

A.2 GAC

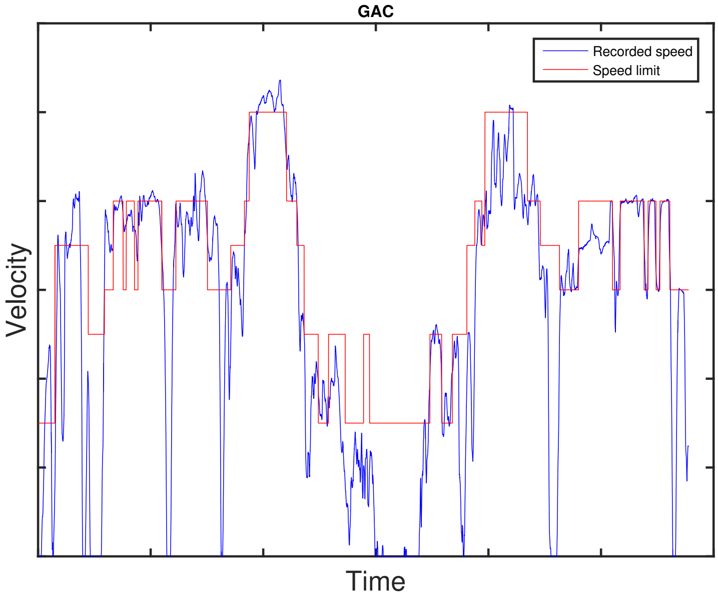


__Figure A.2:__ The speed and speed limits of the GAC cycle.

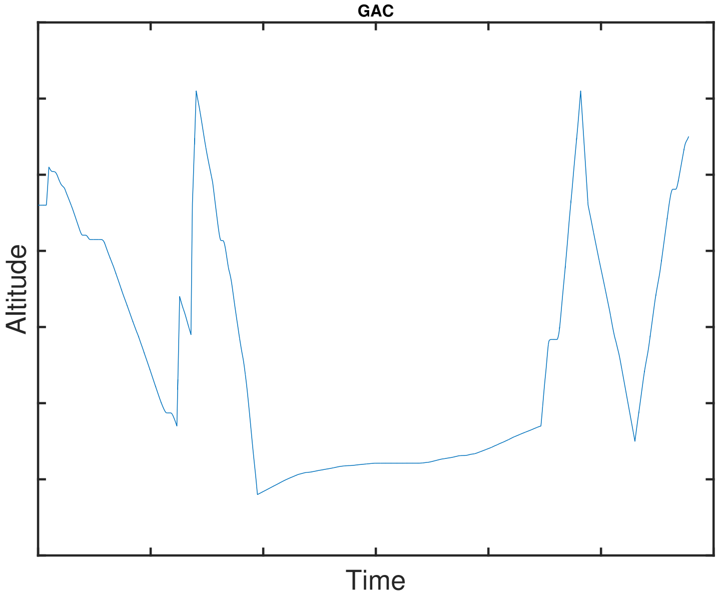


__Figure A.3:__ The altitude profile of the GAC cycle.

A.3 PFC

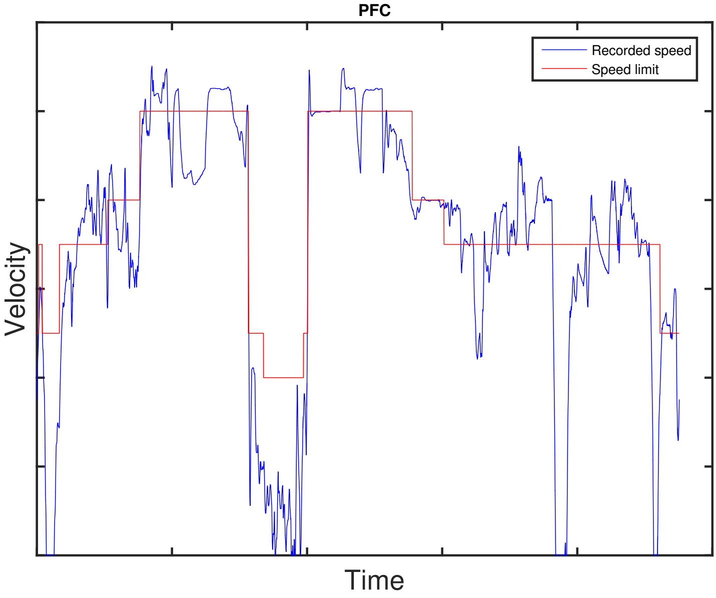


Figure A.4: *The speed and speed limits of the PFC cycle.*

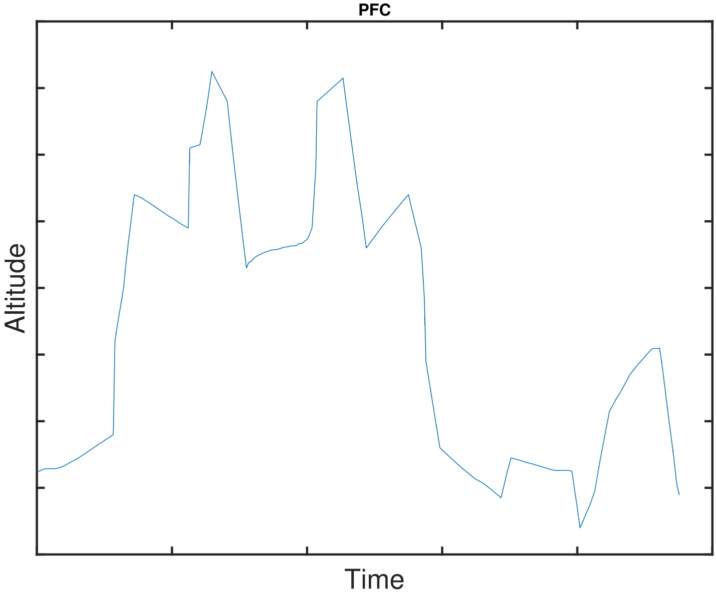


Figure A.5: *The altitude profile of the PFC cycle.*

A.4 WLTP

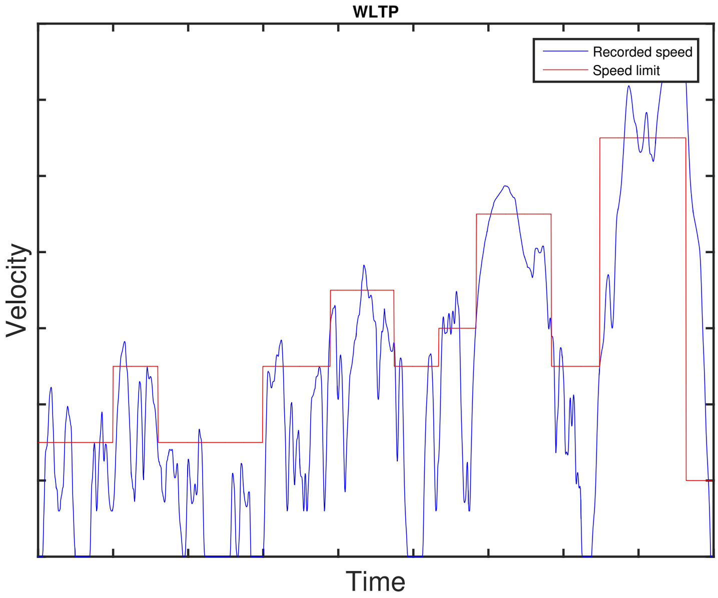


Figure A.6: *The speed of the WLTP Class 3 cycle and estimated speed limits.*

Bibliography

[1] Pei Zhang, Fuwu Yan, and Changqing Du. A comprehensive analysis of energy management strategies for hybrid electric vehicles based on bibliometrics. *Renewable and Sustainable Energy Reviews*, 48:88–104, 2015. Cited on pages 3, 4, and 5.

[2] Sanjaka G Wirasingha and Ali Emadi. Classification and review of control strategies for plug-in hybrid electric vehicless. *IEEE Transactions on Vehicular Technology*, 60(1):111–122, 2011. Cited on page 3.

[3] F. R. Salmasi. Control strategies for hybrid electric vehicles: Evolution, classification, comparison, and future trends. *IEEE Transactions on Vehicular Technology*, 56(5):2393–2404, Sept 2007. ISSN 0018-9545. doi: 10.1109/TVT.2007.899933. Cited on page 3.

[4] Lino Guzzella and Antonio Sciarretta. *Vehicle Propulsion Systems: Introduction to Modeling and Optimization*. Springer, 01 2007. Cited on page 4.

[5] A. Sciarretta, M. Back, and L. Guzzella. Optimal control of parallel hybrid electric vehicles. *IEEE Transactions on Control Systems Technology*, 12(3): 352–363, May 2004. ISSN 1063-6536. doi: 10.1109/TCST.2004.824312. Cited on page 4.

[6] C. Hou, M. Ouyang, L. Xu, and H. Wang. Approximate pontryagin's minimum principle applied to the energy management of plug-in hybrid electric vehicles. *Applied Energy*, 115:174 – 189, 2014. ISSN 0306-2619. Cited on page 4.

[7] Tobias Nüesch, Philipp Elbert, Michael Flankl, Christopher Onder, and Lino Guzzella. Convex optimization for the energy management of hybrid electric vehicles considering engine start and gearshift costs. *Energies*, 7(2):834–856, 2014. ISSN 1996-1073. Cited on pages 4, 25, 39, 43, and 85.

[8] S.Hadj-Said, G.Colin, A.Ketfi-Cherif, and Y.Chamaillard. Convex optimization for energy management of parallel hybrid electric vehicles. *IFAC-*

PapersOnLine, 49(11):271 – 276, 2016. ISSN 2405-8963. 8th IFAC Symposium on Advances in Automotive Control AAC 2016. Cited on page 4.

[9] P. Elbert, T. Nüesch, A. Ritter, N. Murgovski, and L. Guzzella. Engine on/off control for the energy management of a serial hybrid electric bus via convex optimization. *IEEE Transactions on Vehicular Technology*, 63(8):3549–3559, Oct 2014. ISSN 0018-9545. Cited on page 4.

[10] B. Egardt, N. Murgovski, M. Pourabdollah, and L. Johannesson Mardh. Electromobility studies based on convex optimization: Design and control issues regarding vehicle electrification. *IEEE Control Systems*, 34(2):32–49, April 2014. ISSN 1066-033X. Cited on page 4.

[11] L. Serrao, S. Onori, and G. Rizzoni. ECMS as a realization of pontryagin's minimum principle for hev control. In *2009 American Control Conference*, pages 3964–3969, June 2009. Cited on page 5.

[12] Viktor Larsson, Rickard Arvidsson, Anette Westerlund, and Niklas Åkerblom. Dynamometer test of a rule-based discharge strategy for plug-in hybrid electric vehicles. *IFAC-PapersOnLine*, 49:141–146, 12 2016. Cited on pages 5, 27, and 28.

[13] Raimund H.J. Varnhagen and Christian Korthaus. Reduction of fuel consumption with intelligent use of navigation data. In *SAE 2010 Commercial Vehicle Engineering Congress*. SAE International, oct 2010. doi: 10.4271/2010-01-2004. Cited on page 5.

[14] Johan Almgren and Gustav Elingsbo. Route based optimal control strategy for plug-in hybrid electric vehicles. Master's thesis, Linköping University, 2017. Cited on pages 6, 14, 16, 17, 23, 29, 34, 49, 53, and 91.

[15] A. Emadi, Y. J. Lee, and K. Rajashekara. Power electronics and motor drives in electric, hybrid electric, and plug-in hybrid electric vehicles. *IEEE Transactions on Industrial Electronics*, 55(6):2237–2245, June 2008. ISSN 0278-0046. Cited on page 7.

[16] M.F. M. Sabri, K.A. Danapalasingam, and M.F. Rahmat. A comprehensive overview of hybrid electric vehicle: Powertrain configurations, powertrain control techniques and electronic control units. *Energy Conversion and Management*, 52(2):1305 – 1313, 2011. ISSN 0196-8904. Cited on pages 7 and 10.

[17] A. Emadi, K. Rajashekara, S. S. Williamson, and S. M. Lukic. Topological overview of hybrid electric and fuel cell vehicular power system architectures and configurations. *IEEE Transactions on Vehicular Technology*, 54(3): 763–770, May 2005. ISSN 0018-9545. Cited on pages 7, 8, and 10.

[18] A. Sciarretta and L. Guzzella. Control of hybrid electric vehicles. *IEEE Control Systems*, 27(2):60–70, April 2007. ISSN 1066-033X. doi: 10.1109/MCS.2007.338280. Cited on pages 8 and 10.

[19] M.F. M. Sabri, K.A. Danapalasingam, and M.F. Rahmat. A review on hybrid electric vehicles architecture and energy management strategies. *Renewable and Sustainable Energy Reviews*, 53:1433 – 1442, 2016. ISSN 1364-0321. Cited on page 8.

[20] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. doi: 10.1017/CBO9780511804441. Cited on pages 24 and 37.

[21] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, March 2014. Cited on page 24.

[22] LLC CVXGEN. Cvxgen: Code generation for convex optimization. `https://cvxgen.com/docs/index.html`, 2018. Cited on page 24.

[23] A. Domahidi, E. Chu, and S. Boyd. ECOS: An SOCP solver for embedded systems. In *European Control Conference (ECC)*, pages 3071–3076, 2013. Cited on page 24.

[24] embotech. ECOS. `https://www.embotech.com/ECOS`, 2018. Cited on pages 24 and 35.

[25] Kaj Holmberg. *Optimering, Metoder,modeller och teori för linjära, olinjära och kombinatoriska problem*. Liber AB, 2010. Cited on page 25.

[26] D. Ambuhl and L. Guzzella. Predictive reference signal generator for hybrid electric vehicles. *IEEE Transactions on Vehicular Technology*, 58(9):4730–4740, Nov 2009. ISSN 0018-9545. Cited on page 27.

[27] ERTICO. Advancing map-enhanced driver assistance systems. `http://adasis.org/`, 2018. Cited on page 28.

[28] Carl Bredberg and John Stjernrup. Driver modeling, velocity and energy consumption prediction of electric vehicles. Master's thesis, Lund University, 2017. Cited on page 28.

[29] Per Sahlholm, Ather Gattami, and Karl Henrik Johansson. Piecewise linear road grade estimation. In *SAE 2011 World Congress & Exhibition*. SAE International, apr 2011. Cited on page 28.

[30] Stanford University Convex Optimization Group. Qcml: Quadratic cone modeling language. `https://github.com/cvxgrp/qcml`, 2018. Cited on page 37.