Philipp Elbert

NONCAUSAL AND CAUSAL OPTIMIZATION STRATEGIES FOR HYBRID ELECTRIC VEHICLES

Diss. ETH No. 21522

NONCAUSAL AND CAUSAL OPTIMIZATION STRATEGIES FOR HYBRID ELECTRIC VEHICLES

A dissertation submitted to
ETH Zurich

for the degree of
Doctor of Sciences

presented by

Philipp Elbert

MSc ETH in Mechanical Engineering
born June 22, 1982
citizen of Germany

accepted on the recommendation of
Prof. Dr. Lino Guzzella, examiner
Prof. Dr. Theo Hofman, co-examiner

2014

Philipp Elbert
philipp.elbert@alumni.ethz.ch

ETH Zurich
Institute for Dynamic Systems and Control
Sonneggstrasse 3
8092 Zurich, Switzerland

Institute for
Dynamic Systems and Control
IDSC
Institut für Dynamische Systeme
und Regelungstechnik

Für meine Familie

Preface

This work is a product of the AHEAD project, which is a collaboration between Carrosserie HESS AG and the Institute of Dynamic Systems and Control (IDSC) at ETH Zürich, and is sponsored by the Swiss Federal Innovation Promotion Agency (KTI) and the Swiss Federal Office of Energy (BFE).

I am very grateful for being able to participate in this fruitful and exciting research project which kept me motivated and focused throughout the last four years. Many people contributed to this project, and therefore, my sincere gratitude is expressed to the following persons:

Prof. Dr. Lino Guzzella is an ambitious researcher with the ability to motivate his students. During my time at the Institute of Dynamic Systems and Control, he taught me to think like an engineer and to love control systems theory. Prof. Guzzella was my mentor during my bachelor and my master program, and I thank him for giving me the opportunity to work on this exciting research project.

Dr. Chris Onder was my direct contact person throughout the project, and I thank him for his valuable input and never-ending help that ranged from organizing the financial support to assist in finalizing my thesis.

Hans-Jörg Gisler, the technical director of HESS AG, never gave up raising new, unanswered questions and thus helped me to stay curious and interested. I thank him for his initiative during the early phase of the project and his trust when we were implementing new, likely flawed control algorithms on the bus.

Prof. Dr. Theo Hofman and I shared offices for more than six months, and I appreciate our inspiring and delightful discussions. I thank Prof. Hofman for accepting to be the co-examiner of my thesis.

Daniel Ambühl supervised my master student research and he cleared the way for me to join the AHEAD project. Olle Sundström served as a “back-up” supervisor during the very beginning of my doctoral studies. Thank you, Dani and Olle, for your valuable assistance.

Søren Ebbesen's razor-sharp intelligence allows him to conduct excellent research while retaining a great sense of humor. I thank Søren for his cooperation in several joint publications and for making me laugh in both easy and hard times.

Dieter Flubacher and Martin Widmer both joined the AHEAD project several years ago as students. They are still contributing to the project substantially. I thank Dieter and Martin for their initiative to join the project in the first place and for their precious help. Furthermore, I will never forget the intense time we spent together working on the prototype in Heidenheim.

Tobias Nüesch, Andreas Ritter and Nikolce Murgovski co-authored an article with me just recently. Thank you, Tobias, for your valuable inputs and discussions; thank you, Andi, for lending us your impressive programming skills; thank you, Nikolce, for your inspirations in the field of convex optimization.

I thank Ismail Abou-Zeid, Stijn van Dooren, Christian Hohl, Nico Geiser, Dmitri Karpatchev, Duwei Chen, Michael Gugger, Manuel Kant, Christian Muri and Severin Hänggi for the time and effort that they spent contributing to the AHEAD project as interns, bachelor and master students.

My colleagues, were a great support. Thank you for making the last four years as enjoyable as they could be.

Furthermore, I thank the secretarial staff for the well-done administration of the project, and I thank Brigitte Rohrbach for assiduously changing my *denglish* into a more sophisticated language.

Dr. Amrei Wittwer raised my interest in scientific research and encouraged me to actually join the Ph.D. program. Thank you, Amrei, for your inspiration and your care.

My parents and my family always supported me in all the ways imaginable. I humbly thank you, Bettina, Bernd, Julie, Lena and Felix for your never ending support.

January 2014 *Philipp Elbert*

Contents

Abstract

The goal of this thesis is to provide numerical tools to systematically address both the optimal sizing and the control problem of hybrid electric buses. In this context, *sizing* refers to the choice of certain design parameters of the drivetrain, such as the nominal power of the engine, the nominal power of the electric machine, and the nominal energy capacity of the buffer. *Control* relates to the energy management problem, i.e., the decision of how to split the instantaneous power demanded by the driver between the engine and the energy buffer (e.g., battery or supercapacitor). Finally, *optimal* refers to the specific choice of component sizing variables and energy management control inputs that, when combined, globally minimize the fuel consumption of the bus driving along a certain bus line. This thesis focuses on the so-called *serial hybrid electric* or *Diesel-electric* drivetrain.

Part I of this thesis first introduces a mathematical vehicle model and then describes the numerical optimization of the energy management problem with the aim to find the global optimal solution to the problem using all the information that is available, i.e., causal as well as noncausal information. Despite the fact that, due to its noncausality, this globally optimal solution cannot be implemented on a vehicle, it represents an absolute measure that does not depend on any tuning parameters. Therefore, the globally optimal solution serves as a benchmark that allows the potential of a given vehicle design or online energy management to be measured in an objective manner. Two methods are investigated, specifically dynamic programming and convex optimization.

Part II of this thesis describes the development and implementation of a real-time energy management controller whose performance may be termed optimal in the stochastic sense. The control design task is addressed by stochastic dynamic programming that allows a systematic design procedure and facilitates the tuning process required to tune the controller to a behavior that is acceptable in a city bus, e.g., avoiding high noise levels at bus stops or engine shut-offs when the engine is cold. The design methodology is very user friendly, and the resulting controller achieves a close-to-optimal performance on flat inner city bus lines in a practical test. Furthermore, the procedure allows the investigation of the optimal tradeoff between contradicting objectives, such as fuel economy and drivability. In order to reduce any sensitivity towards variations in the type of driving mission, an extension to the baseline controller is developed that takes into account information specific to the bus line, such as the distance to the next stop and the future altitude profile of the bus line. Together with this predictive extension, the baseline controller delivers close-to-optimal results on a hilly bus line as well.

Zusammenfassung

Das Ziel dieser Arbeit ist die Entwicklung systematischer Methoden zur optimalen Dimensionierung von Antriebskomponenten sowie der Entwicklung eines optimalen Energiemanagements für Busse mit seriellem Hybridantrieb. Dimensioniert werden bestimmte Designparameter, wie die nominellen Leistungswerte des Verbrennungs- und des Elektromotors, sowie die Energiespeicherkapazität des elektrischen Speichers (Batterie oder Superkondensator). Mithilfe des Energiemanagements wird die *Lastenaufteilung* bestimmt, d.h., wie die vom Fahrer nachgefragte Leistung zwischen dem elektrischen Energiespeicher und dem Verbrennungsmotor aufgeteilt wird. Die optimale Dimensionierung ist diejenige Kombination von Designparametern und Energiemanagement, welche zusammengenommen eine gewisse Zielfunktion minimieren (z.B. den Treibstoffverbrauch auf einer gewissen Fahrstrecke). Im Zentrum der Betrachtungen steht der *serielle* oder auch *Diesel-elektrische* Hybridantrieb.

Teil I dieser Arbeit bezieht sich auf die numerische Optimierung des Energiemanagements mit dem Ziel, die global optimale Lösung zu finden, wenn nötig unter Berücksichtigung aller verfügbaren Informationen einschliesslich nicht-kausaler Information. Ein solches Energiemanagement kann zwar nicht auf dem Fahrzeug eingesetzt werden, jedoch stellt seine Performance eine absolute Grösse dar, die es erlaubt, verschiedene Auslegungen des Antriebssystems oder des Energiemanagement-Reglers objektiv miteinander zu vergleichen. Zu diesem Zweck werden im ersten Teil der Arbeit zwei Methoden untersucht: die Dynamische Programmierung und die Konvexe Optimierung.

Teil II dieser Arbeit beschreibt die Entwicklung eines Energiemanagement-Reglers für den Einsatz auf dem Fahrzeug. Der Regler minimiert den Treibstoffverbrauch im stochastischen Sinne. Zur Reglerauslegung wurde die stochastische Dynamische Programmierung verwendet. Diese erlaubt eine systematische Reglerauslegung unter Berücksichtigung weiterer Performance-Merkmale, die im Stadtbus nicht vernachlässigt werden

dürfen, z.B. das Vermeiden von hohen Geräuschpegeln an Haltestellen oder das Verhindern eines Start/Stopp-Betriebs des Dieselmotors bei niedrigen Betriebstemperaturen. Diese Methode der Reglerauslegung ist benutzerfreundlich und erreicht im Praxistest ein quasi-optimale Performance auf flachen, innerstädtischen Buslinien. Um auch auf hügeligen Linien optimale Verbrauchswerte zu erzielen, wurde eine Erweiterung des Basis-Energiemanagements entwickelt, welche streckenspezifische Informationen berücksichtigt, z.B., die Distanz bis zur nächsten Haltestelle und das Höhenprofil des weiteren Streckenverlaufs. Zusammen mit dieser prädiktiven Erweiterung erreicht der Energiemanagement-Regler auch auf hügeligen, anspruchsvollen Strecken quasi-optimale Verbrauchswerte.

Introduction

In order to sustain the current level of mobility while at the same time being able to reduce the corresponding CO_2 emissions, we require efficient, economic and comfortable means of transportation. The impact of public transportation systems on our environment might be considered small when it is compared to individual mobility. Yet there is potential to reduce the emissions of noise, pollutants, and CO_2 of public transit systems.

Hybrid electric buses represent one of many promising approaches to reach this goal. The basic idea is to augment the drivetrain of a conventional bus with an electric energy storage system. Such an augmentation potentially reduces fuel consumption, since i) part of the kinetic energy of the bus can be recovered by recuperation and ii) the overall efficiency can be improved by applying intermittent operation of the diesel engine and by moving its operating points towards the most efficient region. On the negative side, electric hybridization increases the weight of the vehicle, the complexity of the drivetrain and the cost. Therefore, only well-designed hybrid electric buses can guarantee a reduction of CO_2 emissions as well as maintain passenger comfort and drivability on an equal level, while being economically competitive with state-of-the-art diesel buses.

The main difficulty encountered during the design is the fact that hybrid electric buses require an energy management system during operation. For instance, if the driver requests an acceleration by pressing down the drive pedal, the energy management system must decide whether the corresponding amount of traction energy is to be delivered by the diesel engine or the energy storage system, or by a combination of both. The quality of this energy management has a significant influence on the energy efficiency of the bus. An energy management strategy that yields good fuel economy for one particular vehicle design A might deliver inferior results when applied to another vehicle design B, even though design B, when equipped with a suited energy management strategy, has the potential to deliver better fuel economy results than design A. Therefore, the optimal sizing problem, i.e., finding the best component sizes for a hybrid electric bus on a given driving mission, is closely coupled with the energy management and cannot be solved separately. The first part of this thesis describes a framework and dedicated numerical methods for finding the optimal design of a hybrid electric bus.

During the operation of a hybrid electric bus, the main difficulty is to ensure optimal fuel economy, no matter what driving situation might occur along a mission. Here, manufacturers mostly rely on heuristic methods and expert intuition. The problem with these methods is that they involve many tuning parameters without any guarantee for an optimality of the results. Furthermore, even though it has been shown that incorporating route information into the energy management can improve its performance, it is very difficult to do so in a heuristic approach. The second part of this thesis presents an energy management controller based on stochastic optimal control which, in a second step, is extended to incorporate route information. Furthermore, experimental results of the implementation of the proposed energy management strategy on a real bus are presented.

Scientific Contribution

Within this research project, several scientific contributions to the literature were published. These are listed below.

Publications i through iii are dedicated to the topic of optimal component design of hybrid electric vehicles. Publication i presents a mathematical vehicle model and the formulation of the energy management problem as an optimal control problem. In publication ii, a numerically efficient implementation of dynamic programming for solving n-dimensional optimal control problems is presented. Publication iii introduces an algorithm based on convex optimization that rapidly solves the energy management problem of a serial hybrid electric bus. Together, the model and the algorithms build the core of the component design software package for HESS AG that was developed over the course of this project. Publication iv summarizes some technical design issues that arose during the course of this project.

Publications v and vi are dedicated to the topic of battery health. Publication v investigates the possibility of improving the lifetime of a battery in hybrid electric passenger cars by augmenting the drivetrain with an additional energy storage comprised of supercapacitors. Publication v proposes an improved energy management strategy that is able to prolong the lifetime of a battery without any further modifications of the vehicle design. Finally, publication vii presents a framework for drivetrain sizing under consideration of several contradictory objectives.

In order to keep this thesis concise, publications v through vii are not discussed explicitly in this text.

i. Elbert, Philipp; Onder, Christopher; Gisler, Hans-Jörg "Capacitors vs. Batteries in a Serial Hybrid Electric Bus" *IFAC Symposium Advances in Automotive Control (AAC)*, 2010, Munich, Germany.

ii. Elbert, Philipp; Ebbesen, Soren; Guzzella, Lino "Implementation of Dynamic Programming for n-Dimensional Optimal Control Problems with Final State Constraints", *IEEE, Transactions on Control Systems Technology*, Vol.21, No.3, pp.924-931, 2013.

iii. Elbert, Philipp; Nüesch, Tobias; Ritter, Andreas; Murgovski, Nikolce; Guzzella, Lino "Optimal Energy Management for a Serial Hybrid Electric Bus via Convex Optimization", Accepted for publication in *IEEE, Transactions on Vehicular Technology*, 2014.

iv. Gisler, Hans-Jörg; Elbert, Philipp "Capacitors vs. Batteries in a Serial Hybrid Electric Bus" *9th Symposium on Hybrid and Electric Vehicles, IAV*, 2012, Braunschweig, Germany.

v. Elbert, Philipp; Ebbesen, Soren; Guzzella, Lino "Economic Viability of Battery Load-Leveling in Hybrid Electric Vehicles using Supercapacitors" *Int. Scientific Conference on Hybrid Electric Vehicles, RHEVE*, 2011, Paris, France.

vi. Ebbesen, Soren; Elbert, Philipp; Guzzella, Lino "Battery State-of-Health Perceptive Energy Management for Hybrid Electric Vehicles" *IEEE, Transactions on Vehicular Technology*, Vol.61, pp.2893-2900, 2012.

vii. Ebbesen, Soren; Elbert, Philipp; Guzzella, Lino "Engine Downsizing and Electric Hybridization Under Consideration of Cost and Drivability" *Oil & Gas Science and Technology – Rev. IFP Energies Nouvelles*, Vol.67, pp.109-116, 2012.

Structure of this Thesis

This text is divided in two main parts. Part I is based on publications i, ii and iii and presents a mathematical vehicle model and two algorithms that rapidly evaluate the optimal energy management problem of hybrid electric buses and thus allow the optimal component sizes for a given vehicle on a given mission to be found. Part II discusses the design of an online energy management strategy and its implementation on a serial hybrid electric bus, and it presents the corresponding experimental results.

Part I

Optimal Component Design

Chapter 1

Vehicle Modeling

This chapter introduces a mathematical model of a serial hybrid electric bus that was built by HESS AG in 2012. The model will later be used for simulations and controller synthesis. The modeling methodology is adopted from [1], resulting in a quasi-static model with a sampling frequency of 1 Hz. The model was first presented in publication i, while some extensions are taken from publication iii.

1.1 The Vehicle

Figure 1.1 shows a photograph of the vehicle, which is a 12 m city bus with a total capacity of 85 passengers. The drivetrain components are highlighted in Figure 1.2. The vehicle is propelled by an electric traction motor (blue) of 280 kW, while the energy needed for propulsion is delivered by a 170 kW diesel engine generator unit (red). Additionally, a 625 Wh supercapacitor with a peak power of 300 kW (green) is used to buffer electric energy, e.g., from brake energy recuperation. The energy management is implemented in a dedicated on-board electronic control unit (yellow).

1.2 Longitudinal Vehicle Dynamics

Figure 1.3 shows the forces that act on the vehicle body during driving. Assuming that the vehicle speed $v(t)$ and acceleration $a(t)$, the road grade $\alpha(t)$ and the number of passengers on board $n_p(t)$ have all been recorded along a mission, the required traction force $F_t(t)$ induced by the driving cycle is given by

$$F_t(t) = (m(t) + m_{\text{rot}}) \cdot a(t) + F_a(t) + F_r(t) + F_g(t), \tag{1.1}$$

Figure 1.1: Photograph of a serial hybrid electric bus.

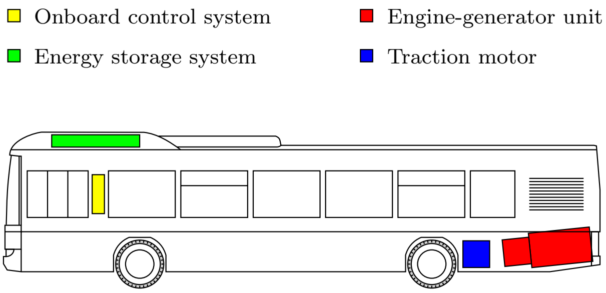


Figure 1.2: Drivetrain components of a serial hybrid electric bus.

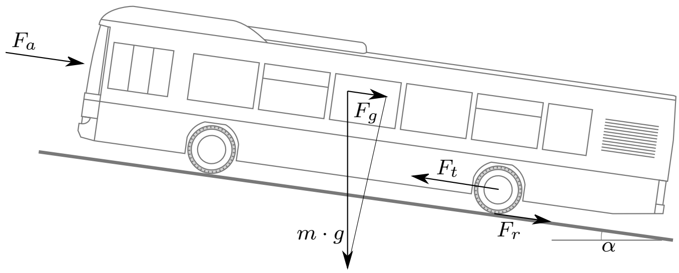


Figure 1.3: Forces acting on the vehicle body during driving.

where $F_a(t)$ is the aerodynamic drag force, $F_r(t)$ the rolling friction force, and $F_g(t)$ is the up/downhill driving force given by

$$F_a(t) = 0.5 \cdot \rho_a \cdot c_d \cdot A \cdot v^2(t), \tag{1.2}$$
$$F_a(t) = m(t) \cdot g \cdot c_r \cdot \cos(\alpha(t)), \tag{1.3}$$
$$F_g(t) = m(t) \cdot g \cdot \sin(\alpha(t)). \tag{1.4}$$

The model accounts for forces in longitudinal direction only. All latitudinal forces, variations of friction parameters during curves, wind forces, and other disturbances are neglected. The term

$$m_{rot} = \frac{n_w \Theta_w + \Theta_m \gamma_t^2}{r_w^2} \tag{1.5}$$

accounts for the overall inertia of the wheels $n_w \Theta_w$ and for that of the traction machine Θ_m. The mass of the vehicle can be approximated a function of the number of passengers on board $n_p(t)$

$$m(t) = m_v + n_p(t) \cdot 75\,\text{kg}, \tag{1.6}$$

where m_v is the mass of the vehicle excluding passengers. When sizing the drivetrain components of the vehicle, the change in vehicle weight has to be taken into account. Therefore, the vehicle weight is modeled as a function of the weights of the drivetrain components

$$m_v = m_0 + m_m + m_g + m_b, \tag{1.7}$$

where m_0 stands for the mass of the vehicle without the powertrain components, and m_m, m_g, and m_b stand for the masses of the traction motor, the engine-generator unit and the energy buffer, respectively.

Table 1.1: Vehicle Model Parameters.

Parameter	**Symbol**	**Value**	**Unit**
Gravitation constant	g	9.81	[kg·m/s^2]
Air density	ρ_a	1.20	[kg/m^3]
Empty vehicle mass	m_v	13.6	[t]
Equivalent rotational mass	m_{rot}	0.6	[t]
Frontal area	A	8.2	[m^2]
Drag coefficient	c_d	0.9	[–]
Rolling friction coefficient	c_r	0.65	[%]
Number of wheels	n_w	4	[–]
Wheel radius	r_w	0.466	[m]
Transmission ratio	γ_t	9.82	[–]
Transmission efficiency	η_t	0.98	[–]

The electric traction motor is connected to the wheels via a single gear and a final drive with an overall transmission ratio of γ_t. Thus, the resulting torque, rotational speed and acceleration at the traction motor are given by

$$\omega_m(t) = \gamma_t \cdot v(t)/r_w, \qquad \dot{\omega}_m(t) = \gamma_t \cdot a(t)/r_w, \tag{1.8}$$

and

$$T_m(t) = \begin{cases} \frac{F_t(t)\cdot r_w}{\gamma_t \cdot \eta_t} & \text{for } F_t(t) \geq 0 \\ \max\{\frac{F_t(t)\cdot r_w \cdot \eta_t}{\gamma_t}, T_{m,\min}(\omega_m(t))\} & \text{for } F_t(t) < 0, \end{cases} \tag{1.9}$$

where the overall transmission efficiency η_t is assumed to be constant. The wheels have an effective radius of r_w. The second case of (1.9) stands for the possibility that kinetic energy has to be dissipated using the friction brakes if the traction motor regenerative torque limit $T_{m,\min}(\omega_m(t))$ (explained in next subsection) is exceeded.

Numerical values of the chassis model parameters are given in Table 1.1.

1.3 Traction Motor

The electric traction motor is modeled by a scalable consumption map. The electric power drawn (in traction mode) or supplied to the DC link (in recuperation mode) is given by

$$P_m(t) = \Gamma_m(w_m(t), T_m(t)), \tag{1.10}$$

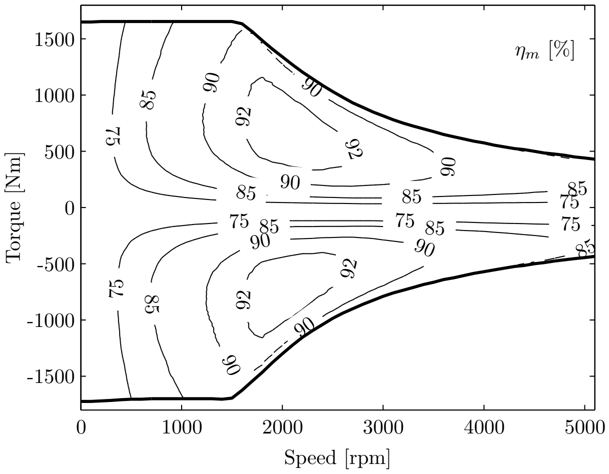


Figure 1.4: Efficiency map of an electric traction machine of 280 kW.

where Γ_m is the electricity consumption map of the machine. The map includes modeled energy losses in cables and in the converter. The traction torque is limited by a speed dependent upper and lower bound given by

$$T_{m,\min}(\omega_m) \leq T_m \leq T_{m,\max}(\omega_m). \tag{1.11}$$

Figure 1.4 shows the efficiency map and the speed dependent torque limits of the electric traction machine of 280 kW used in the vehicle. The data was provided by the manufacturer.

1.4 Power Split

The traction motor, the generator, the buffer, the braking resistor and the vehicle auxiliaries are connected electrically via a DC link that allows energy to be distributed among the components. Since the storage capacity of the DC link itself is negligible, the following power balance holds

$$P_g(t) + P_b(t) \geq P_{\text{req}}(t), \tag{1.12}$$

where $P_g(t)$ stands for the electric power delivered by the generator and $P_b(t)$ for the buffer power. The inequality stands for the possibility to dissipate electric energy in a braking resistor that is used only if the power request is negative and the energy buffer is fully charged, or if the negative power request exceeds the charging power limitation of the buffer.

The power request consists of the electric traction power and the auxiliary power which was recorded along the driving mission together with the vehicle speed

$$P_{\text{req}}(t) = P_m(t) + P_{\text{aux}}(t). \tag{1.13}$$

Note that for a given vehicle configuration and a given driving mission, the power request is fully defined and can therefore be calculated prior to the optimization of the energy management strategy. If $P_{\text{req}}(t)$ is available from a measurement on the vehicle, the data can be directly used as an input to the drivetrain model. Inside the energy management strategy, the power request is considered a disturbance.

1.5 Engine Generator Unit

The engine generator unit (EGU) consists of a diesel engine that is mechanically connected to a generator via a crankshaft and a dedicated control system. Given a desired electric power to be supplied to the DC link, the EGU control system has to i) define a rotational speed setpoint that allows the engine to deliver enough mechanical power without stalling, ii) control the rotational speed to that setpoint and iii) control the engine and generator torque setpoints in such a way that the actual electric power matches the desired power.

The engine rotational speed can be chosen freely by the control system and thus represents a degree of freedom that can be used to optimize the conversion efficiency from fuel to electricity. Applying a static optimization algorithm results in an optimal operating curve that relates the electric output power with an optimal rotational speed.

Figure 1.5 shows the overall conversion efficiency from fuel to electricity including the DC converter as a function of speed and torque. The optimized operating curve (thin black line) is comprised of several distinct operating points (black dots). The most efficient operating point is found at 90 kW. Note that the DC converter reduces its frequency below 1300 rpm resulting in a better conversion efficiency, which explains the discontinuity of the iso-efficiency lines. In the range from 50 to 90 kW, the operating points were all shifted to 1200 rpm. While this procedure com-

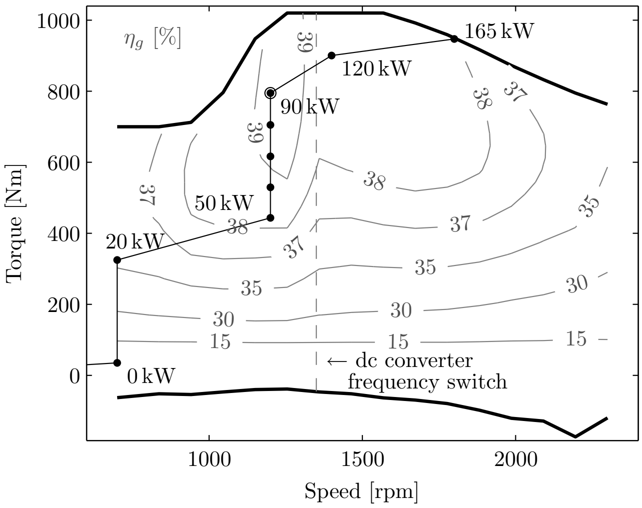


Figure 1.5: Efficiency map of a 165 kW engine generator unit. The optimized operating points are indicated by black dots

promises the overall efficiency to a small extent, it allows for a smoother operation of the engine generator unit in the range between 50 and 90 kW, which is visited frequently in practice. The maximum rotational speed is limited to 1800 rpm where the engine already delivers peak power. Avoiding rotational speeds above 1800 rpm helps to reduce noise levels in the bus.

The quasi static model assumes that the transition between any two operating points on the optimal operating curve can be executed within one second and thus within one time step of the model. Therefore, the fuel energy consumed can be modeled as a static function of the electric output power

$$P_f(t) = \dot{m}_f(P_g(t)) \cdot H_{\text{lhv}} \cdot e(t), \tag{1.14}$$

where $\dot{m}_f(P_g(t))$ is obtained from interpolation in a lookup table that has been determined by a static optimization involving the consumption maps of the engine and the generator. The variable H_{lhv} denotes the lower heating value of diesel fuel. The binary decision variable $e(t) \in \{0, 1\}$ is required to represent the engine on/off decision. The electric output power $P_g(t)$ is limited to $P_g(t) \in [0, P_{g,\max} \cdot e(t)]$.

Obviously, such a simplified model neglects many dynamic phenomena, such as the additional fuel needed during an engine start/stop, the reduced engine torque during transients due to the dynamics of the turbo charger, etc. Therefore, the model presented here is only useful for vehicle design. In the context of online energy management, as described in Chapter 4, a different model able to capture those effects will be introduced.

1.6 Energy Buffer

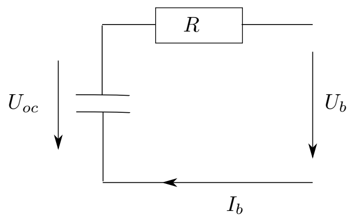


Figure 1.6: Supercapacitor equivalent circuit model.

The energy buffer, which can be either a battery or a supercapacitor, is represented using an equivalent circuit model, i.e, a voltage source $U_{oc}(t)$ in series with a resistance R (see Figure 1.6). The power delivered to the DC link $P_b(t)$ is described by

$$P_b(t) = U_{oc}(t) \cdot I_b(t) - R \cdot I_b^2(t), \tag{1.15}$$

where $I_b(t)$ is the current flowing through the buffer. Solving (1.15) for the current yields

$$I_b(t) = \frac{1}{2R}\left(U_{oc}(t) - \sqrt{U_{oc}^2(t) - 4 \cdot P_b(t) \cdot R_b}\right), \tag{1.16}$$

where the internal resistance R is assumed to be constant. Per definition, a positive current (and positive power) will discharge the buffer and therefore

$$\frac{d}{dt}E(t) = -U_{oc}(t) \cdot I_b(t). \tag{1.17}$$

The energy content of the buffer at any time t is given by

$$E(t) = \int_0^{Q(t)} U_{oc}(t)\; dQ. \tag{1.18}$$

The absolute value of the current flowing through the energy buffer is limited to a maximum value, leading to the following power limitations

$$\begin{aligned} P_{b,\max}(t) &= I_{b,\max}U_{oc}(t) - R_b I_{b,\max}^2 & (1.19)\\ P_{b,\min}(t) &= -I_{b,\max}U_{oc}(t) + R_b I_{b,\max}^2. & (1.20)\end{aligned}$$

The nominal power is defined as $P_{b,\text{nom}} = P_{b,\max}$.

1.6.1 Supercapacitor

The vehicle under consideration is equipped with a bank of supercapacitor modules. In this case, the open circuit voltage is a linear function of the charge present on the buffer

$$U_{oc}(t) = \frac{Q_{sc}(t)}{C_{sc}}. \tag{1.21}$$

Table 1.2 summarizes the most important parameters of the supercapacitor stack used in the vehicle.

Table 1.2: Parameters of the supercapacitor.

Parameter	**Symbol**	**Value**	**Unit**
Capacity	C_{sc}	12.5	[F]
Internal resistance	R_{sc}	100	[mΩ]
Max. voltage	$U_{sc,\max}$	600	[V]
Peak current	$I_{sc,\max}$	500	[A]
Mass	m_{sc}	250	[kg]
Peak power	$P_{sc,\max}$	300	[kW]
Energy capacity	$E_{sc,\text{nom}}$	625	[Wh]

Chapter 2

Dynamic Programming

Dynamic Programming (DP) is a powerful numerical method for solving optimal control problems [2, 3]. For example, it can be used to calculate the optimal energy management strategy for a hybrid vehicle on a given driving mission. The main advantage compared to other methods is the guarantee for a global optimal solution—regardless of the type of problem. The dynamic system can be non-linear, non-smooth, and it can include both continuous and discrete variables. The most notable drawback of dynamic programming is the *curse of dimensionality*, i.e., the fact that the computational effort grows exponentially with the number of state variables and inputs of the underlying dynamic system. Another potential problem is that the dynamic programming algorithm requires perfect a priori knowledge of the entire driving cycle, and thus the solution is non-causal.

Before applying the dynamic programming algorithm, all continuous variables need to be discretized. It is possible to circumvent discretization, e.g., if the so-called cost-to-go function can be expressed analytically. However, in most cases, discretization is the only viable option. Thus, the accuracy of the proposed solution depends on the integrity of the grid, i.e., the spacing between each node in the discretized variables. In order to moderate numerical errors arising from discretization, a careful implementation is crucial.

If a problem includes *final* state constraints, any solution trajectory is bound to lie inside the backward-reachable space, i.e., the space from which the final state constraint can be reached within the given final time. The backward-reachable space represents a dynamic state constraint that neither coincides with the state constraints of the problem definition, nor with the grid that is used for discretization. It was shown in [4] that it is important to carefully distinguish between backward-reachable and non-backward-reachable space. Otherwise, significant numerical errors may distort the solution.

In this chapter, a seemingly simple, yet non-trivial and efficient implementation of the basic dynamic programming algorithm combined with the reachability theory developed by [5, 6] and [7] is proposed. The algorithm avoids numerical errors that are due to the interpolation between backward-reachable and non-backward-reachable grid points, which improves the accuracy of the found solution significantly. A case study is used to demonstrate how the proposed method may lower the computational effort by over 300 times without loss of accuracy. In contrast to that, previously proposed methods to reduce the computational effort of dynamic programming, such as iterative [8], adaptive [9], approximate [10] neuro- [11] or state increment dynamic programming [12], either sacrifice global optimality or are applicable to a specific class of problems only. Publications [4] and [13] propose algorithms that are based on the same basic idea of calculating the backward-reachable space. However, they are only applicable to first order systems, whereas the algorithm proposed here is applicable to systems of any order.

The proposed algorithm has been integrated into the dpm-function [14], which can be downloaded under `www.idsc.ethz.ch/Downloads`.

This chapter is based on publication ii, which has been co-authored by Søren Ebbesen.

2.1 Optimal Control Problem

At this stage, only optimal control problems with fixed final time and partially constrained final states are considered. The system has n state variables and m inputs. The state variables are assumed to be continuous variables. The proposed method can also handle problems where some, or all variables are discrete, however, the benefits of the proposed method are only apparent when the optimal control problem contains at least one continuous state variable. The underlying dynamic system can either be a continuous-time or a discrete-time system.

The optimal control problem is summarized as follows: find an admissible control sequence u_k, $k = 0, 1, \ldots, N$ such that the cost functional (2.1) is minimized and the constraints (2.2–2.6) are satisfied.

$$\min_{u_k \in U_k} \left\{ g_N(x_N) + \sum_{0}^{N-1} g_k(x_k, u_k) \right\} \tag{2.1}$$

$$\begin{aligned} x_{k+1} &= f_k(x_k, u_k) && (2.2)\\ x_k &\in X_k \subseteq \mathbb{R}^n && (2.3)\\ x_0 &= x_{\text{init}} && (2.4)\\ x_N &\in T \subseteq \mathbb{R}^n && (2.5)\\ u_k &\in U_k \subseteq \mathbb{R}^m && (2.6)\\ & \text{for all } k = 0, 1, \ldots, N. \end{aligned}$$

The function $g_N(x)$ is the final cost term and $g_k(x, u)$ is the stage-cost, i.e., the cost of applying the control signal u at discrete time k to the dynamic system given by (2.2). Note that g_k and f_k are allowed to be time-variant, hence the index k. The state variables are constrained to the time-variant set X_k. The initial condition is given by x_{init} and the final value is partially constrained by the target set T. Additionally, the input signals are constrained by the time-variant set U_k.

Due to the discrete nature of dynamic programming, it is necessary to discretize the independent variable, i.e., time, the state space and the control space. Therefore, the functions f_k and g_k are discrete-time representations of the dynamic system and the stage-cost function. At time k the state space is discretized to the set $X_k = \{x_k^1, x_k^2, \ldots, x_k^q\}$. The superscript i in x_k^i denotes the state variable in the discretized state-time space at the node with time index k and state index i. The continuous state vector is denoted by x_k. Analogously, the control space is represented by the discrete set $U_k = \{u_k^1, u_k^2, \ldots, u_k^r\}$. Recall that the control inputs can be either discrete or continuous. In the latter case U_k is a discrete approximation of the true control space.

Typically, the final state constraint defined by (2.5) cannot be met starting from every point in the search space. The evolution of the backward-reachable space and its boundary is depicted in Figure 2.1 for a one dimensional system. In this case, the exact calculation of the backward-reachable space with the use of model inversion techniques, as was done in [13] and [4] does not pose any particular problem. However, in the case of a two dimensional problem, Figure 2.2, the the backward-reachable space at any time $k \neq N$ might be non-convex, even if the target set T is convex.

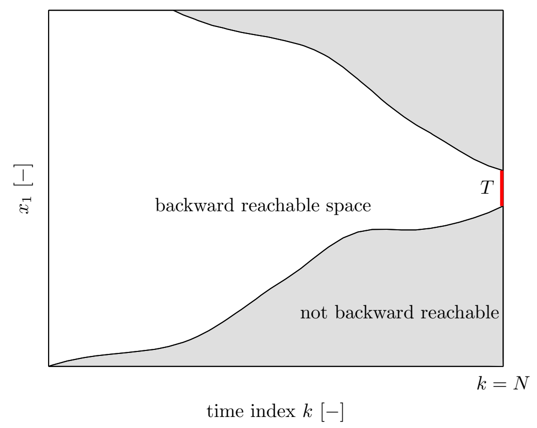


Figure 2.1: Backward-reachable space for a system with only one dynamic state variable. The final state constraint is given by the set T.

This makes the numerical representation of the backward-reachable space a difficult task—especially in higher dimensional problems.

2.2 Basic Dynamic Programming

When applying dynamic programming, the optimal cost-to-go function $\mathcal{J}_k(x^i)$ is evaluated at every node in the discretized time-state space by proceeding backwards in time, according to the following algorithm:

1. Initialization of cost-to-go function:

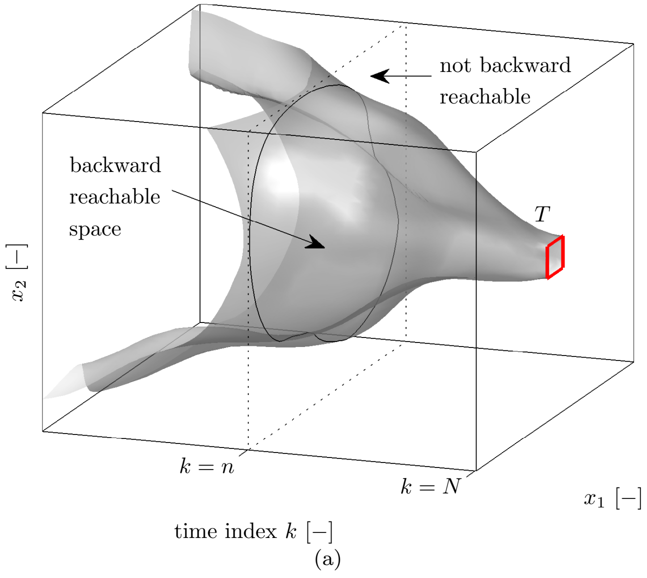
$$\mathcal{J}_N(x^i) = \begin{cases} g_N(x^i) & \text{for } x^i \in T \\ \infty & \text{else.} \end{cases} \tag{2.7}$$

2. Backward iteration for $k = N - 1$ to 0, $\forall x^i \in X_k$:

$$\mathcal{J}_k(x^i) = \min_{u_k \in U_k} \left\{ g_k(x^i, u_k) + \mathcal{J}_{k+1}(f_k(x^i, u_k)) \right\}. \tag{2.8}$$

The optimal control signal at each node is given by the argument minimizing the right-hand side of (2.8), yielding the optimal control policy $\pi = \{\mu_0(x), \mu_1(x), \ldots, \mu_{N-1}(x)\}$.

Grid points that are not backward-reachable, are of course infinitely expensive and therefore should have infinite cost, as in (2.7). However,



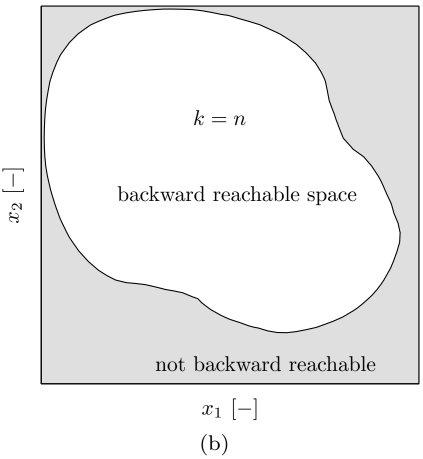


Figure 2.2: (a) Backward-reachable space for a system with two dynamic state variables. (b) Note that in this example, the backward-reachable space at time $k = n$ is a non-convex set.

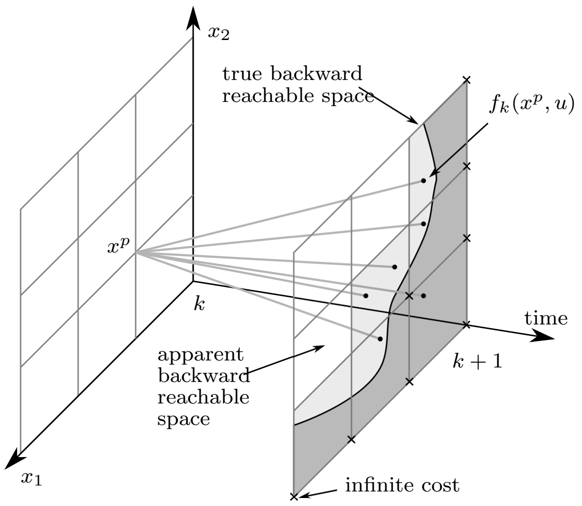

x_2
true backward reachable space
$f_k(x^p, u)$
x^p
k
time
$k+1$
apparent backward reachable space
x_1
infinite cost

Figure 2.3: Interpolation between grid points with finite and infinite costs diffuses the boundary of the backward-reachable space. The light gray area is backward-reachable, but will apear not to be.

this causes numerical problems. Consider the scenario depicted in Figure 2.3. The cost-to-go $\mathcal{J}_{k+1}(x)$ is known for all grid points x^i at time $k+1$. In order to evaluate the cost-to-go at point x^p at time k, the algorithm simulates the system over one time step by applying all possible control candidates. Thereby, the system is driven into the points $f_k(x^p, u)$ with $u \in U_k$. Since these points do not generally coincide with the grid, interpolation is used to find the values of the cost-to-go at $\mathcal{J}_{k+1}(f_k(x^p, u))$. In this study we used multi-linear interpolation. In the example shown in Figure 2.3, interpolation will always rely on at least one grid point where the cost-to-go has an infinite value. Therefore, the backward-reachable space will appear smaller than it actually is and the cost-to-go at $\mathcal{J}_k(x^p)$ will be set to infinity, even though x^p is perfectly backward-reachable. This effect propagates throughout the entire grid, and thus the calculated backward-reachable space will be underestimated.

A first remedy to this problem is to choose a large but finite cost instead of an infinite value in (2.7). However, this technique results in a steep gradient in the cost-to-go function that distorts the found solution.

This procedure will be referred to as basic dynamic programming. Other techniques would be to increase the density of the grid towards the end of the problem, either by increasing the number of grid points (increase q, as k approaches N), or decreasing the size of the search space. It will be shown later that these methods fail to produce useful results.

2.3 Representation of the Backward-Reachable Space

A prerequisite for understanding the improved dynamic programming algorithm, which will be introduced in the next section, is to understand how level-set functions can be used to calculate the backward-reachable space [5],[6],[7].

Generally, in an n-dimensional state space, it is not clear how the boundaries of the backward-reachable space evolve. Furthermore, at any time k, this space is not necessarily convex (see Figure 2.2). Therefore, an exact numerical description is difficult. However, the feasible region at each time k can be conveniently estimated by an implicit surface function or level-set function, which is calculated by means of dynamic programming. The main idea is the following. Let $\mathcal{I}$ be a function that acts on $\mathbb{R}^n$:

$$\mathcal{I} : X \subseteq \mathbb{R}^n \to \mathbb{R}. \tag{2.9}$$

Such a function can be used to represent a region G that is defined as follows:

$$G = \{x \in X | \mathcal{I}(x) \leq 0\}. \tag{2.10}$$

Figure 2.4 illustrates how a level-set function $\mathcal{I}(x)$ can be used to represent the backward-reachable space shown in Figure 2.2. The advantages of such a representation are the following. Firstly, a Cartesian grid can be used for the evaluation of the function $\mathcal{I}(x)$. Together with (2.10) it is numerically easy (by interpolation) to evaluate whether a point x is backward-reachable or not. Secondly, such a function can represent regions of any shape (even non-convex). Furthermore, the description is general and can be applied to systems with any number of state variables and control inputs. A drawback of this method is that the boundary of the backward-reachable space is not represented exactly, but rather approximated by the level-set function. The error of this approximation decreases with increasing grid density.

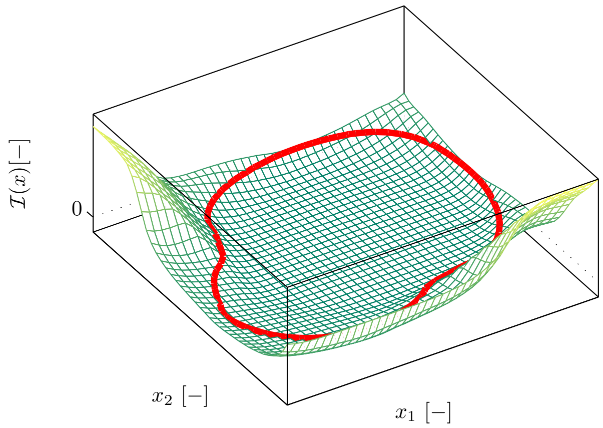


Figure 2.4: Example of a level-set function $\mathcal{I}(x)$ at some time index k. The bold curve represents $\mathcal{I}(x) = 0$, i.e., the boundary of the backward-reachable space.

2.3.1 Computing the Backward-Reachable Space

Assume that the final state constraint is given as a target set T, which is defined by a level-set function $h(x)$:

$$h : X_N \to \mathbb{R}, \quad \text{where } X_N \subseteq \mathbb{R}^n \tag{2.11}$$

$$T = \{x \in X_N | h(x) \leq 0\}. \tag{2.12}$$

Then, a dynamic programming algorithm is applied:

1. Initialization:
$$\mathcal{I}_N(x^i) = h(x^i). \tag{2.13}$$

2. Backward iteration for $k = N - 1$ to 0:
$$\mathcal{I}_k(x^i) = \min_{u_k \in U_k} \left\{ \mathcal{I}_{k+1}(f_k(x^i, u_k)) \right\}. \tag{2.14}$$

The cost-to-go function of this algorithm is interpreted as a level-set function. Therefore, a new symbol $\mathcal{I}$ is used in order to distinguish it from the cost-to-go of the original problem. Figure 2.5 illustrates the algorithm. Assume that the level-set function $\mathcal{I}_{k+1}(x)$ at time $k + 1$ is known at

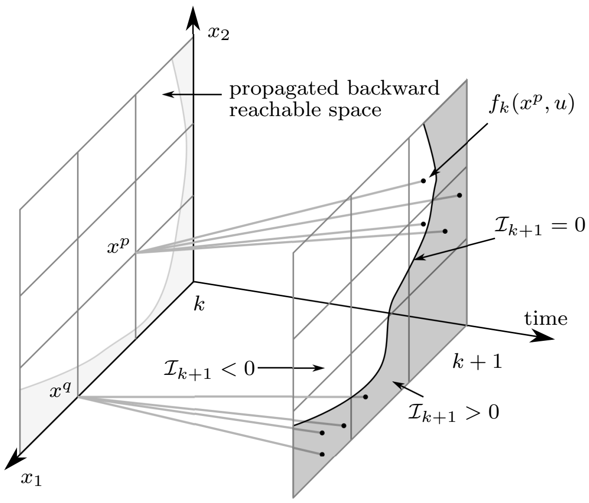


Figure 2.5: Illustration of the algorithm that propagates the backward-reachable space for a system with two dynamic state variables.

all nodes x^i. Starting from a specific grid point x^p at time k, the algorithm applies all possible control candidates. This process yields the points $f_k(x^p, u)$ with $u \in U_k$. Then, as in (2.14), the algorithm assigns $\mathcal{I}_k(x^p)$ to be the minimum forward reachable value of $\mathcal{I}_{k+1}$, i.e., the minimum value of $\mathcal{I}_{k+1}(f_k(x^p, u))$. In the example, this value is negative for x^p, which indicates that x^p is a backward-reachable grid point. Contrarily, the point x^q is not backward-reachable.

2.4 The Improved Dynamic Programming Algorithm

The main idea behind the improved dynamic programming algorithm is to use a level-set function to describe the backward-reachable space. This allows identifying the grid points from which the final state constraints cannot be met. Thus, large penalties in the cost-to-go function are no longer necessary and the inherent steep gradient is avoided.

The value of the cost-to-go function at grid points outside the backward-reachable space is not defined. Since interpolation might have to rely on such grid points, a reasonable approximation needs to be found.

The improved algorithm simply uses the value of the cost of driving the system parallel to the boundary of the backward-reachable space, as close as possible to the final state constraint. This delivers a smooth cost-to-go function over the whole state space and therefore, interpolation will not cause any particular numerical problem. This algorithm is referred to as level-set algorithm.

2.4.1 The Level-Set Algorithm

1. Initialize $k = N$ and the level-set and the cost-to-go functions as

$$\begin{aligned} \mathcal{I}_N(x^i) &= h(x^i) && (2.15) \\ \mathcal{J}_N(x^i) &= g_N(x^i), && (2.16) \end{aligned}$$

   where $h(x)$ is chosen to represent the target set T as in (2.11) and (2.12).

2. Then reduce k by one and update the level-set function by

$$\mathcal{I}_k(x^i) = \min_{u_k \in U_k} \left\{ \mathcal{I}_{k+1}(f_k(x^i, u_k)) \right\}. \tag{2.17}$$

3. For each grid point x^i, find the set of control signals for which the system trajectory ends up inside the backward-reachable space in the next time step

$$U_k^{\mathrm{F}}(x^i) = \{ u_k \in U_k | \mathcal{I}_{k+1}(f_k(x^i, u_k)) \leq 0 \} \tag{2.18}$$

   and the one control candidate that minimizes the level-set function

$$\tilde{u}_k(x^i) = \underset{u_k \in U_k}{\operatorname{argmin}} \left\{ \mathcal{I}_{k+1}(f_k(x^i, u_k)) \right\}. \tag{2.19}$$

4. Update the optimal cost-to-go by the following rule: if at least one valid control candidate is found, i.e., $U_k^{\mathrm{F}}(x^i) \neq \emptyset$, then calculate the cost-to-go based upon the optimal candidate

$$\mathcal{J}_k(x^i) = \min_{u_k \in U_k^{\mathrm{F}}(x^i)} \left\{ g_k(x^i, u_k) + \mathcal{J}_{k+1}(f_k(x^i, u_k)) \right\}. \tag{2.20}$$

   If, however, the grid point is not backward-reachable, then calculate the cost-to-go based on the control input $\tilde{u}_k(x^i)$

$$\mathcal{J}_k(x^i) = g_k(x^i, \tilde{u}_k(x^i)) + \mathcal{J}_{k+1}(f_k(x^i, \tilde{u}_k(x^i))). \tag{2.21}$$

   and repeat steps 2-4 until $k = 0$.

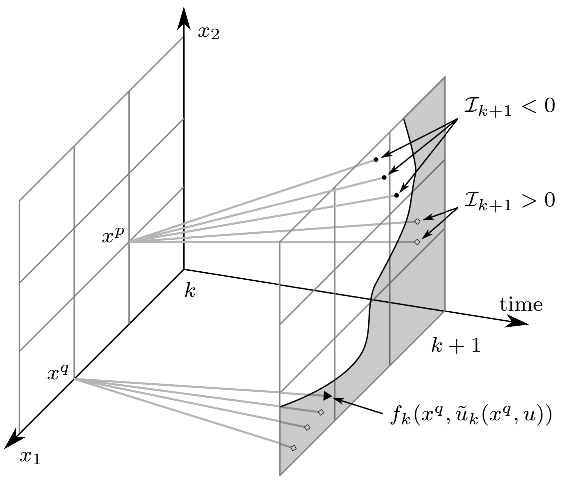


Figure 2.6: Illustration of the level-set DP algorithm for a system with two state variables.

Consider the scenario depicted in Figure 2.6. At time $k+1$ the cost-to-go and the level-set functions are known, e.g., from step 1. For the point x^p at time k, the algorithm applies all possible control candidates. This will drive the system to the points $f_k(x^p, u)$ with $u \in U_k$. Then, (step 2), the value of the level-set function $\mathcal{I}_k$ at point x^p is set to be the minimum reachable value of $\mathcal{I}_{k+1}$, which will propagate the information about backward reachability to time k. Then (step 3) the algorithm identifies the valid control candidates $U_k^F(x^p)$, based on the values of the level-set function at the points $\mathcal{I}(f_k(x^p, u))$ (black dots vs. empty diamonds). Finally (step 4), the cost-to-go at point x^p is calculated considering the valid control inputs only, as given by Eq. (2.20). The level-set function thus enables the algorithm to distinguish between those trajectories that end up inside the backward-reachable set and those who do not. Therefore, a large penalty term in the cost-to-go to avoid infeasible states is no longer needed, and numerical errors arising from the interpolation between a large penalty and actual cost-to-go values are avoided.

After all, interpolation may still have to rely on grid points that are not part of the backward-reachable space. For such a grid point x^q, not a single valid control input can be found that drives the system back into the backwards reachable space. In this case, the value of the cost-to-go function is computed based on the control $\tilde{u}_k(x^q)$ as in (2.21). This control input drives the system as closely as possible towards the backward-reachable space (black triangle), which yields a cost-to-go function that is smooth over the whole state space.

2.4.2 Forward Simulation

The result of the dynamic programming algorithm is the optimal cost-to-go function for all times k and all grid points x_k^i. In order to retrieve the sequence of optimal control inputs and the corresponding optimal state trajectory, a forward simulation is performed. For the level-set algorithm, this forward simulation takes the following form:

1. Initialization at $k = 0$:
$$x_0 = x_{\text{init}}. \tag{2.22}$$

2. Find the feasible control candidates
$$U_k^{\mathrm{F}} = \{u_k \in U_k | \mathcal{I}_{k+1}(f_k(x_k, u_k)) \leq 0\}. \tag{2.23}$$

3. If at least one valid control candidate can be found, i.e., if $U_k^{\mathrm{F}} \neq \emptyset$, find the optimal control input by
$$u_k^o(x_k) = \underset{u_k \in U_k^{\mathrm{F}}}{\operatorname{argmin}} \left\{ g_k(x_k, u_k) + \mathcal{J}_{k+1}(f_k(x_k, u_k)) \right\}. \tag{2.24}$$
If, however, the point x_k is not backward reachable then use the control input given by
$$u_k^o(x_k) = \underset{u_k \in U_k}{\operatorname{argmin}} \left\{ \mathcal{I}_{k+1}(f_k(x_k, u_k)) \right\}. \tag{2.25}$$

4. Simulate the system using the optimal control input:
$$x_{k+1} = f_k(x_k, u_k^o, w_k) \tag{2.26}$$
and repeat steps 2, 3 and 4 until $k = N$.

Note that in the forward simulation x_k can take continuous values, therefore the superscript i does not appear in (2.22)-(2.26).

Equation (2.25) is not strictly necessary, however it improves the robustness of the algorithm. It becomes active if x_k comes to lies outside the feasible region only. However, if x_{init} is inside the feasible region, this case remains inactive.

2.4.3 Initialization of the Target Set

It is often desired to reach a specific final state, i.e., a single point in the state space at time $k = N$. However, this is practically impossible to achieve because only a discrete number of control signals are available. Instead, it is advisable to accept a certain tolerance. In other words, a target set, rather than a target point, is specified

$$T = [x_1^{\min}, x_1^{\max}] \times \ldots \times [x_n^{\min}, x_n^{\max}] \in \mathbb{R}^n, \tag{2.27}$$

where $x_n^{\min}$ and $x_n^{\max}$ are the minimum and maximum allowed values for the n-th state variable. When using linear interpolation, it makes sense to define the function $h(x)$ such that its value at any point x is equal to the distance from x to the nearest bound

$$h(x) = \max_{j=1,\ldots,n} \left\{ \max(x_j^{\min} - x_j, x_j - x_j^{\max}) \right\}. \tag{2.28}$$

Note that the sign of $h(x)$ is such that

$$h(x) \begin{cases} \leq 0 \text{ if } x \in T \\ > 0 \text{ else.} \end{cases} \tag{2.29}$$

Thus the minimum of $h(x)$ resides in the interior of the target set T.

2.4.4 Initialization of the Cost-to-go Function

As k approaches the final time N, the optimal trajectory is likely to move along the boundary of the backward-reachable space and thus the interpolation needs to rely on grid points that are not part of the backward-reachable space. Recall that the value of the cost-to-go function outside backward-reachable space is calculated based on an approximation—the control input that drives the system as close as possible to the target set T. Therefore, the final cost function needs to take into account the cost of driving the system from a point $x^i \notin T$ into T. A linear approximation to replace (2.16) can easily be found and is sufficient in most cases

$$\mathcal{J}_N(x^i) = \begin{cases} g_N(x^i) & \text{if } x^i \in T \\ g_N(x^i) + \lambda^T \cdot (x^i - x_T) & \text{else.} \end{cases} \tag{2.30}$$

Thereby $(x^i - x_T)$ is the minimum distance from point x^i to the target set T and λ is a vector of factors that converts the distance between the grid point x^i and the target set T into an equivalent cost. In literature the vector λ is typically referred to as equivalence factor or Lagrange multiplier, hence the symbol λ.

2.4.5 Computational Effort

Typically, the number of model evaluations is the factor that has the greatest influence on calculation time. The number of model-function evaluations for the basic DP and the level-set DP with an equally spaced grid is given by:

$$N_{\text{feval}} = N \cdot \prod_{i=1}^{n} N_{x_i} \cdot \prod_{j=1}^{m} N_{u_j} \tag{2.31}$$

where N_{x_i} is the number of grid points in the i-th state variable and N_{u_j} represents the number of grid points in the j-th control input.

Compared to the basic dynamic programming implementation, the level-set algorithm leads to some increase in the computational demand due to the interpolation of the level-set function (2.17) and the determination of the valid control candidates (2.18). However, the system dynamic equation f_k and the arc-cost g_k have to be evaluated only once for each state and control input combination. In the limit case, where the time to evaluate the model function tends to zero, the total evaluation time doubles compared to that required by the basic algorithm. However, in a typical problem, the evaluation of the model itself accounts for the largest fraction of the calculation time. Thus, when using either algorithm to solve the identical problem using the same grid discretization, the calculation time for the level-set algorithm is only slightly longer than for the basic approach.

2.5 Case Study: Simple Dynamic System

The accuracy of a solution obtained by dynamic programming is highly dependent on the discretization of the state space. Therefore, choosing a good level of discretization is a trade-off between accuracy and calculation time. If a finer grid is chosen, the solution becomes more precise, but also calculation time increases.

In this section, a simple optimal control problem based on a dynamic system with two state variables and two inputs is used to compare the accuracy of the basic DP and the level-set DP algorithms. Since the analytic solution to the problem is known, a fair comparison of the results is possible.

2.5.1 Problem Definition

The system is described by the following dynamic equations:

$$\dot{x}_1(t) = -\frac{1}{2}x_1(t) + u_1(t)\cdot u_2(t) \tag{2.32}$$

$$\dot{x}_2(t) = -\frac{1}{2}x_2(t) + u_1(t)\cdot (1 - u_2(t)) \tag{2.33}$$

with the constraints

$$x(t) \in [0,1]\times[0,1] \quad \forall t \in [0,t_f] \tag{2.34}$$

$$u(t) \in [0,1]\times[0,1] \quad \forall t \in [0,t_f] \tag{2.35}$$

and the initial and final conditions

$$x_1(0) = x_2(0) = 0 \tag{2.36}$$

$$x_1(t_f) = x_2(t_f) \geq 0.5 = x_{\min} \tag{2.37}$$

where $t_f = 2$ seconds. The cost functional to be minimized is given as:

$$J = \int_0^{t_f} u_1(t) + 0.1\cdot|u_2(t) - 0.5|dt. \tag{2.38}$$

The problem can be illustrated as in Figure 2.7. Within time t_f, two water reservoirs have to be filled from empty to a certain level $x(t_f) = [0.5, 0.5]^T$, using the minimum amount of water. But the reservoirs both have a leak, where the outflow is proportional to the amount of water in the reservoir. The first control input determines the total water flow into the system, whereas the second input determines how the water is distributed between the two reservoirs. The second term in the cost functional penalizes the action on the second control input and ensures that the problem has a unique solution.

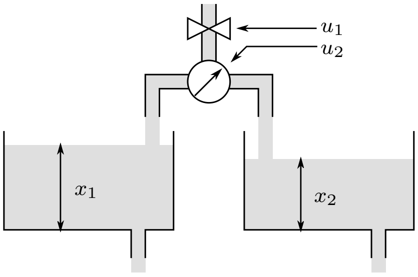


Figure 2.7: Illustration of the example problem.

The optimal solution to this problem is to start filling in the water as late as possible at time t_{on} with the maximum inflow, such that the final state constraint is exactly met at time t_f. The inflow has to be distributed between the two reservoirs equally. This way, the amount of water lost through the leaks is minimized over the time horizon. Therefore, the optimal control input is given as:

$$u_1^o(t) = \begin{cases} 0 & \text{if} \quad t < t_{on} \\ 1 & \text{if} \quad t \geq t_{on} \end{cases} \tag{2.39}$$

$$u_2^o(t) = 0.5 \quad \forall t \in [0, t_f] \tag{2.40}$$

The optimal time to switch on the water is calculated by backwards integrating the system dynamics from the final state constraint using the optimal control inputs. It can be shown that

$$t_{on} = 2 + 2 \cdot \ln\left(\frac{1}{2}\right). \tag{2.41}$$

Therefore, the value of the optimal cost functional is

$$\begin{aligned} J^o_{\text{analytic}} &= \int_{t_{on}}^{t_f} 1 \, dt = t_f - t_{on} \qquad (2.42) \\ &= -2 \cdot \ln\left(\frac{1}{2}\right) \approx 1.3863. \end{aligned}$$

Since the optimal solution is to move along the boundary of the feasible region, the dynamic programming algorithm is sensitive to the method used to account for the final state constraints.

2.5.2 Resolution Study

The problem stated above is solved using the basic and level-set DP algorithms using various levels of state space discretization. The time discretization is chosen to be $\Delta t = 0.01$ seconds. The number of points used to discretize each state variable are $N_x = N_{x_1} \cdot N_{x_2}$, while the control space discretization is kept at $N_u = N_{u_1} \cdot N_{u_2} = 21 \times 21 = 441$. Due to the discretization of time, an inherent minimal error is introduced

$$\frac{J^o_{\text{DP}} - J^o_{\text{analytic}}}{J^o_{\text{analytic}}} = \frac{1.39 + 2 \cdot \ln(\frac{1}{2})}{-2 \cdot \ln(\frac{1}{2})} = 0.0027. \tag{2.43}$$

For the basic dynamic programming algorithm the penalty cost for non backward-reachable states is chosen to be the maximum cost that can occur in the problem. The maximum cost accumulates when the water valve is opened at $t = 0$ seconds and not closed again before t_f, and therefore $\mathcal{J}^\infty = \int_0^{t_f} 1 dt = 2$.

For the level-set algorithm the cost-to-go function was initialized using (2.30)

$$\mathcal{J}_N(x^i) = \sum_{n=1}^{2} \max(x_{\min} - x^i_n, 0), \tag{2.44}$$

where x^i_n is the n-th element of the vector $x^i \in X_N$. This accounts for the fact that if either one of reservoirs n ends up at time $k = N$ with a level lower than $x_{\min}$, it has to be filled up to the level $x_{\min}$ in future, thus causing a future cost of $(x_{\min} - x^i_n)$. From a physical standpoint it therefore makes sense that the vector of equivalence factors is $\lambda = [1, 1]^T$ in this example.

Figure 2.8 shows the found system trajectory and the corresponding control inputs, when the algorithms are used with $N_x = 51 \times 51$. In the upper graph the optimal trajectories of $x^o_1(t)$ and $x^o_2(t)$ are equal and therefore coincide. Clearly, the solution found by the level-set algorithm is much closer to the analytic solution. With the basic approach, the water valve is opened much too early and therefore more water is lost through the leaks. The system is not able to move closely along the boundary of the backward-reachable space due to the numerical problems described before.

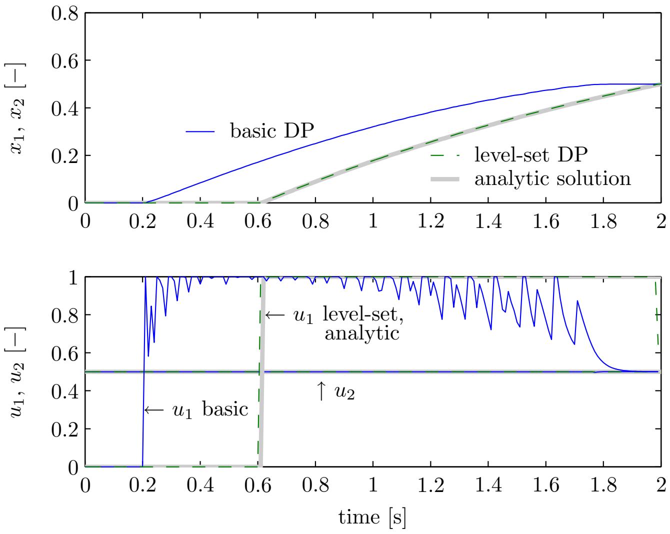


Figure 2.8: System trajectory and control inputs when solving the problem with $N_x = 51 \times 51$.

The corresponding control inputs are depicted in the lower graph of Figure 2.8. It can be seen that the basic algorithm causes a very erratic trajectory for the first control input. The level-set method has the additional benefit that the control inputs are much smoother.

The relative deviation of the cost computed by dynamic programming compared to the results obtained by the analytic solution is shown in Figure 2.9. Over the whole range of levels of discretization the results obtained by the level-set algorithm are substantially closer to the analytic solution than those of the basic dynamic programming approach. Even at the lowest level of discretization, the level-set algorithm yields better results than the basic approach at the highest level of discretization.

When looking at the results presented in Figure 2.9 it is immediately clear that an adaptive grid method, based on refining discretization towards the end of the problem, can never reach the accuracy of the proposed algorithm. E.g., an algorithm starting with $N_x = 11 \times 11$ and ending with $N_x = 201 \times 201$ will never be more accurate than an algorithm with a constant $N_x = 201 \times 201$. Obviously, the new proposed method delivers superior results.

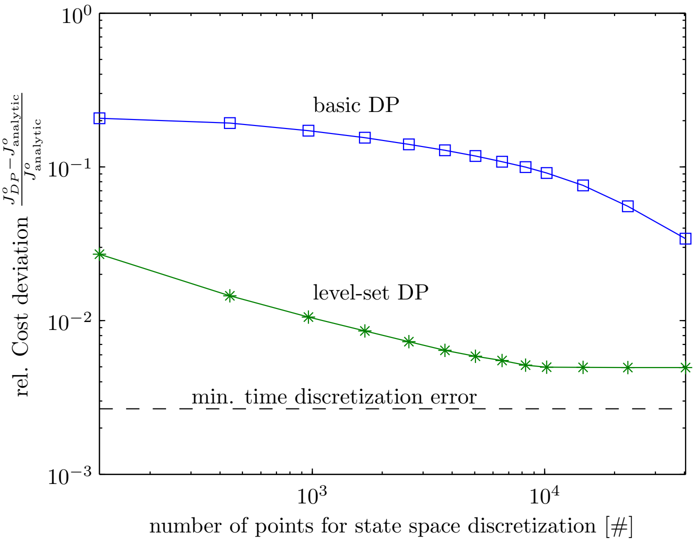


Figure 2.9: The relative deviation of the cost computed by DP compared to the optimal cost obtained by the analytic solution for several levels of discretization ranging from $N_{x_1} = N_{x_2} = 11$ up to 201.

2.5.3 Computational Effort

The computational effort of the level-set and basic dynamic programming algorithms is shown in Table 2.1 for a similar level of accuracy. The level-set algorithm allows choosing a significantly lower level of discretization. Instead of a grid with 201 $\times$ 201 points, a grid of 11$\times$11 points is sufficient. Assuming that the speed of the algorithm is mainly determined by the number of function evaluations, as defined in (2.31), the time savings can be calculated, with the result that the level-set DP algorithm is 334 times faster than the basic DP approach, and yet the resulting cost is closer to that of the analytic solution.

In order to consolidate this statement, the calculation time to solve the problem on a 2.66 GHz processor was measured. The last row in Table 2.1 shows that the evaluation time is accelerated by a factor of about 390 when using the level-set DP algorithm instead of the basic DP algorithm. This result seems somewhat counterintuitive since the benefit of lowering the discretization is even greater than the value that was esti-

Table 2.1: Results obtained with the two algorithms at similar accuracy for the simple dynamic system problem.

		basic DP	**level set DP**
level of discretization	N_x	201 × 201	11 × 11
relative cost deviation	ΔJ	0.034	0.027
function evaluations	N_{feval}	3'581'185'041	10'725'561
measured eval. time	t_{eval} [min]	164	0.42

mated using (2.31). The reason for this effect is that the handling of the large amounts of data, as required for a high level of discretization, causes an additional significant computational effort which was not accounted for in (2.31). Working with a low level of discretization therefore has an additional positive effect.

2.6 Conclusion

The level-set algorithm presented in this paper considerably improves the efficiency of dynamic programming. The method is generic as it does not impose any additional requirements on the properties of the optimal control problem and it does not compromise the global optimality of the found solution. The method can handle non-linear optimal control problems with any number of state variables and control inputs. Of course, the method can only improve the efficiency of the basic algorithm if the underlying dynamic system contains at least one continuous state variable. In fact, many real-world optimal control problems faced in engineering belong to this class. In such problems the optimal state trajectory is close to the boundary of the backward-reachable space when final time approaches. Computational cost can be reduced since the proposed algorithm achieves the same accuracy as the basic implementation at a much lower state space resolution.

Chapter 3

Convex Optimization

Hybrid electric vehicles represent a promising approach to reduce both fuel consumption and CO_2 emissions. However, the system complexity makes the design and control of such vehicles a difficult task. In component sizing, i.e., finding the best possible size of the components in the drivetrain, the globally optimal energy management should be used in order to exclude the influence of a possibly sub-optimal energy management [15]. One method to calculate the globally optimal energy management is dynamic programming [2, 3]. While being successfully applied to hybrid vehicles [16, 17, 18, 19], dynamic programming is computationally expensive. The so-called curse of dimensionality limits the application of dynamic programming to low-order systems of typically not more than one or two state variables. Another method to solve the energy management problem is based on Pontryagin's minimum principle [20]. The theory has been applied to hybrid vehicles [21, 22] where Pontryagin's optimality conditions lead to a two-point boundary value problem. However, the high sensitivity towards the initial condition and the presence of state constraints make it difficult to solve the problem.

In order to avoid these drawbacks, researchers recently proposed using convex optimization [23] to solve the optimal energy management problem [24, 25, 26, 27, 28]. Using interior point convex solvers, the optimal solution is obtained in polynomial time [23]. Moreover, convex optimization allows to optimize design parameters and the energy management simultaneously [29]. Unfortunately, discrete decision variables, such as the engine on/off decision or the gear selection, cannot be included in a convex formulation. The resulting problem would be a mixed-integer problem and the computational effort to solve it would increase exponentially with its

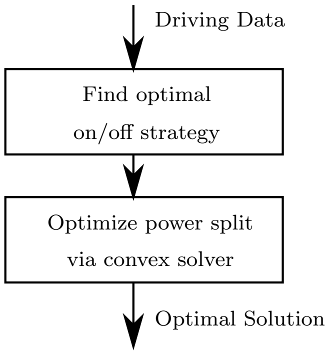

Driving Data
Find optimal on/off strategy
Optimize power split via convex solver
Optimal Solution

Figure 3.1: Sequential optimization of engine on/off strategy and power split.

duration.[1] Currently available mixed-integer solvers can handle problems of very short duration only, e.g. they are useable in a receding horizon control problem [25]. In the case of hybrid vehicle optimization, however, much longer time horizons are needed. Thus, solving a mixed-integer problem is not suitable.

A more effective technique consists of splitting the optimization problem into two parts [27, 28], as illustrated in Figure 3.1. The first part consists of defining an engine on/off strategy, while the second part consists of optimizing the power split based on that on/off strategy. This procedure delivers optimal results if and only if the pre-defined engine on/off strategy is optimal. In [30] the procedure is repeated several times to improve the initial on/off strategy, but after all, the strategy is heuristic and is not able to guarantee global optimality.

In this chapter, the optimal engine on/off strategy is calculated analytically using Pontryagin's minimum principle. In Section 3.2, the optimal strategy is to switch the engine on if and only if the requested power is above a certain threshold [31, 32] that is a function of the energy level of the buffer and of the equivalence factor. This on/off control law is then used together with convex optimization to iteratively find the optimal solution. Thereby, the algorithm makes use of information about the equivalence

[1]E.g. the number of possible solutions to a problem with only one binary decision variable and a horizon of n time steps is proportional to 2^n.

factor that is provided by the convex solver in form of a dual variable. The algorithm is then shown to converge towards a solution that fulfills the necessary conditions for an optimal solution. In two case studies, the proposed algorithm is shown to produce optimal results in less time than the dynamic programming.

This chapter is structured as follows: In Section 3.1, the energy management problem is formulated along with a description of the vehicle model. In Section 3.2, the optimal engine on/off conditions are derived analytically. Moreover, the iterative algorithm to calculate the optimal engine on/off strategy is explained in more detail. In Section 3.3, the methods presented are validated in two case studies.

This chapter is based on publication iii, which has been co-authored by Tobias Nüesch, Andreas Ritter and Nikolce Murgovski.

3.1 Convex Problem Formulation

The following sections summarize the steps required to obtain a quasi-static [1] convex vehicle model. Since the steps required to obtain a convex model, especially the relaxation of equality constraints by inequalities, are analogous to [29, 33], these steps are described only briefly. The numerical parameters of the vehicle model are summarized in Table 1.1.

In a series configuration, all disturbances to the energy management can be lumped into a single electrical power request

$$P_{\text{req}}(t) = P_m(t) + P_{\text{aux}}(t),$$

which can be pre-calculated for a given vehicle on a given mission. Furthermore, if the component sizes are not included into the problem formulation as a decision variable, it is not necessary to reformulate the models of the vehicle dynamics and the electric traction machine in convex form. In fact, it has been shown that such a reformulation is possible without too strong simplifications or approximations [29].

3.1.1 Convex EGU Model

In order to arrive at a convex model, it is necessary to approximate the EGU consumption map by a convex expression, i.e., a second order polynomial

$$P_f(t) = (p_2 \cdot P_g^2(t) + p_1 \cdot P_g(t) + p_0) \cdot e(t) \tag{3.1}$$

Table 3.1: Parameters of the engine-generator unit model.

Parameter	Symbol	Value	Unit
Engine-generator power	$P_{g,\max}$	165	[kW]
Polynom parameter	p_0	16.1	[kW]
Polynom parameter	p_1	2.22	[-]
Polynom parameter	p_2	0.0013	[1/kW]

where $e(t) \in \{0, 1\}$ is a binary decision variable required to represent the engine on/off decision. The electric output power $P_g(t)$ is limited to $P_g(t) \in [0, P_{g,\max} \cdot e(t)]$.

Figure 3.2 shows the conversion efficiency from fuel to electric energy along the optimal operating curve. Clearly, the error introduced by the approximation is small throughout the operating range. The parameters of the convex engine generator model are summarized in Table 3.1.

3.1.2 Convex Energy Buffer Model

The buffer is represented by an equivalent circuit consisting of a voltage source $U_{oc}(t)$ connected in series with a resistance R. The power delivered to the DC link $P_b(t)$ is described by

$$P_b(t) = U_{oc}(t) \cdot I_b(t) - R \cdot I_b^2(t), \tag{3.2}$$

where $I_b(t)$ is the current in the buffer. Solving (3.2) for the current yields

$$I_b(t) = \frac{1}{2R}\left(U_{oc}(t) - \sqrt{U_{oc}^2(t) - 4RP_b(t)}\right), \tag{3.3}$$

where the term beneath the square-root is nonnegative. This can be concluded by maximizing P_b in (3.3) with respect to I_b, which gives

$$P_b(t) < \frac{U_{oc}^2(t)}{4R}. \tag{3.4}$$

The internal resistance R is assumed to be a constant [1], and the open-circuit voltage is assumed to be an affine function of the charge $Q(t)$ of the buffer

$$U_{oc}(t) = \frac{Q(t)}{\tilde{C}} + U_0. \tag{3.5}$$

The parameter U_0 has the unit of a voltage. The parameter $\tilde{C}$ has the unit of a capacitance and is thus denoted as *equivalent capacitance*. Figure 3.3

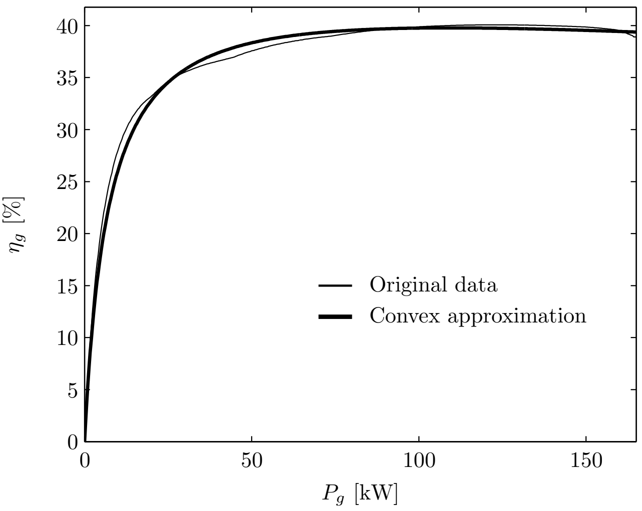


Figure 3.2: Original and approximated efficiency along the optimal operating curve of a 165 kW engine generator unit as a function of the electric output power P_g. Parameters of the convex model are listed in Table 3.1.

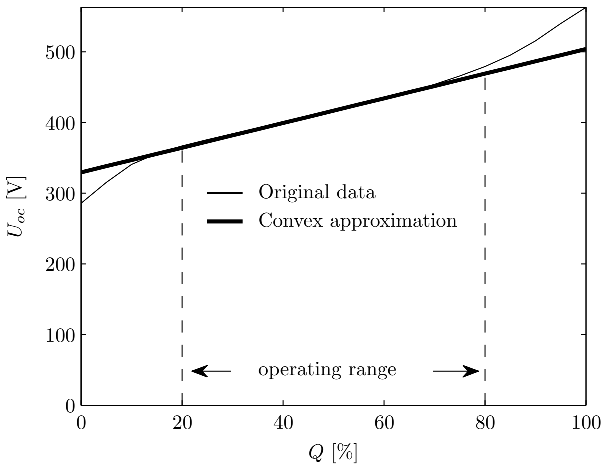


Figure 3.3: Original and approximated open-circuit voltage curve of a 25 kWh battery, as a function of the battery state of charge. The parameters for the convex model are listed in Table 3.3.

shows the original and the approximated open-circuit voltage curve of a 25 kWh battery pack as a function of $Q(t)$. Clearly, the affine approximation holds well inside the operating range.

In order to obtain a convex model, the buffer energy $E(t)$ is chosen as the state variable. The energy content at time t is calculated by

$$E(t) = \int_0^{Q(t)} U_{oc}(y)\, dQ \tag{3.6}$$

$$= \int_0^{Q(t)} (\frac{1}{\tilde{C}} Q(t) + U_0)\, dQ \tag{3.7}$$

$$= \left(\frac{Q^2(t)}{2\tilde{C}} + U_0 Q(t) \right) \tag{3.8}$$

By solving (3.5) for $Q(t)$, and inserting the result into (3.8), the following relationship is found

$$E(t) = \frac{1}{2}\tilde{C} \cdot (U_{oc}^2(t) - U_0^2) \tag{3.9}$$

$$= \frac{1}{2}\tilde{C} U_{oc}^2(t) - E_0, \tag{3.10}$$

with $E_0 = \frac{1}{2}\tilde{C}U_0^2$. Note that in the case of a supercapacitor $U_0 = 0\,\text{V}$, and thus $E_0 = 0\,\text{J}$.

The equation for the buffer dynamics is found by realizing that

$$\frac{dE(t)}{dt} = U_{oc}(t) \cdot \frac{Q(t)}{dt} = U_{oc}(t) \cdot I_b(t). \tag{3.11}$$

In (3.11), $U_{oc}(t)$ and $I_b(t)$ can be replaced using (3.10) and (3.3), respectively. These steps yield the following dynamic equation

$$\frac{dE(t)}{dt} \leq -\frac{1}{R\tilde{C}} \Bigg[E(t) + E_0 - \sqrt{(E(t) + E_0)(E(t) + E_0 - 2R\tilde{C}P_b(t))} \Bigg]. \tag{3.12}$$

The right-hand side of (3.12) is concave because it consists of the sum of an affine function and the geometric mean of two non-negative affine functions[2]. Note that (3.12) has been relaxed by an inequality, which is

[2]The term $E(t) + E_0 - 2R\tilde{C}P_b(t)$ cannot become smaller than zero because of (3.4).

necessary to make the model convex. In the case of the optimal solution, (3.12) will hold with equality. The proof of this claim is straightforward. Assume that the convex solver finds the optimal solution and (3.12) holds with inequality. This means that the variation of the buffer energy in a given time interval does not correspond to the cumulative input/output power of the buffer, and hence some energy was wasted. Clearly, a better solution can be found ((3.12) holding with equality). This in turn means that the initial assumption ((3.12) holding with inequality is optimal) was wrong.

The energy content of the battery is constrained to the interval

$$E(t) \in [E_{\min}, E_{\max}] , \tag{3.13}$$

where $E_{\min}$ and $E_{\max}$ are obtained by inserting the minimum and maximum allowed charge $Q(t)$ into (3.10). Moreover, the buffer power minimum and maximum power limitations are approximated by affine functions in $E(t)$,

$$P_b(t) \geq b_{u,1} \cdot E(t) + b_{u,0}, \tag{3.14}$$

$$P_b(t) \leq b_{l,1} \cdot E(t) + b_{l,0}. \tag{3.15}$$

Details on the derivation of convex buffer model can be found in [33].

3.1.3 Optimal Control Problem

The optimal control problem is formulated as follows. For each time instance from $t = 0$ to t_f, find the optimal buffer power $P_b^o(t)$ that minimizes the cost criterion

$$J(P_b(t)) = \int_0^{t_f} P_f(P_b(t))\, dt \tag{3.16}$$

subject to the charge conservation condition $E(t_f) = E(0) = E_{\text{init}}$ and the vehicle model described above.

The complete convex optimal control problem is given by (3.17) in Table 3.2. Note that, due to its non-convex nature, the engine on/off decision $e(t)$ cannot be included as a decision variable. That decision must precede the process of solving the convex problem. Problem (3.17) corresponds to the second block in Fig. 3.1, and its solution can be globally optimal if and only if the pre-defined engine on/off strategy $e(t)$ is globally optimal.

Table 3.2: Convex subproblem

Minimize

$$\sum_{k=1}^{N} \left(p_2 \cdot \boldsymbol{P}_g^2(k) + p_1 \cdot \boldsymbol{P}_g(k) + p_0\right) \cdot e(k) \cdot \Delta t \tag{3.17a}$$

subject to

$$\boldsymbol{P}_g(k) \geq P_{\text{req}}(k) - \boldsymbol{P}_b(k) \tag{3.17b}$$

$$\boldsymbol{P}_g(k) \leq P_{g,\max} \cdot e(k) \tag{3.17c}$$

$$\boldsymbol{P}_g(k) \geq 0 \tag{3.17d}$$

$$\begin{aligned}\boldsymbol{E}(k+1) \leq \boldsymbol{E}(k) - \frac{\Delta t}{R\tilde{C}} \Bigg[& \boldsymbol{E}(k) + E_0 \\ & - \sqrt{\boldsymbol{E}(k) + E_0} \cdot \sqrt{\boldsymbol{E}(k) + E_0 - 2R\tilde{C}\boldsymbol{P}_b(k)} \Bigg]\end{aligned} \tag{3.17e}$$

$$\boldsymbol{P}_b(k) \leq b_{u,1} \cdot E(k) + b_{u,0} \tag{3.17f}$$

$$\boldsymbol{P}_b(k) \geq b_{l,1} \cdot E(k) + b_{l,1} \tag{3.17g}$$

$$\boldsymbol{E}(k) \leq E_{\max} \tag{3.17h}$$

$$\boldsymbol{E}(k) \geq E_{\min} \tag{3.17i}$$

$$\boldsymbol{E}(1) = \boldsymbol{E}(N+1) = E_{\text{init}} \tag{3.17j}$$

for all $k \in [1, \ldots, N]$.

The variable N denotes the number of time steps Δt $(= 1\,\text{s})$ along the driving mission. Bold symbols represent free optimization variables.

3.2 Derivation of the Optimal On/Off Strategy

In this section, Pontryagin's minimum principle is used to derive the optimal engine on/off strategy based on a simplified buffer model. With this simplification, the optimal engine on/off strategy is shown to be a function of the requested power, the equivalence factor, and buffer energy level only.

3.2.1 Simplified Buffer Model

The power losses P_l over the internal resistance of the energy buffer

$$P_l(t) = RI^2(t) \tag{3.18}$$

$$= \frac{\left(U_{oc}(t) - \sqrt{U_{oc}^2(t) - 4RP_b(t)}\right)^2}{4R} \tag{3.19}$$

can be approximated well by a second-order polynomial. Applying a Taylor series expansion around $P_b = 0$ leads to

$$P_l(t) = \frac{R}{U_{oc}^2(t)} \cdot P_b^2(t) = \frac{\tilde{C} \cdot R}{2(E(t) + \Delta E)} \cdot P_b^2(t) = \beta(t) \cdot P_b^2(t). \tag{3.20}$$

Figure 3.4 shows the original and the approximated losses of a 25 kWh battery as a function of the battery power for a fully charged and a fully discharged battery. It can be seen that the approximation holds reasonably well. Note that the approximations introduced in this section are used to calculate the engine on/off strategy only. All simulations conducted later are based on the convex model introduced in Section 3.1.

3.2.2 Simplified Vehicle Model

For fixed vehicle parameters, the required power $P_{\text{req}}(t)$ can be pre-calculated and thus, the overall equations of the simplified model can be summarized

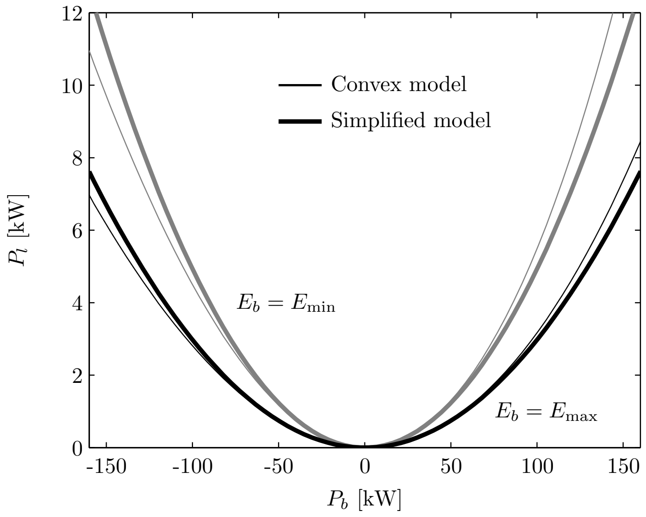


Figure 3.4: Convex and simplified power loss model of a 25 kWh battery, as a function of the battery power for a fully charged and fully discharged battery. The parameters for the convex model are listed in Table 3.3.

as follows:

$$P_{\text{req}}(t) \leq P_g(t) + P_b(t), \tag{3.21a}$$

$$P_f(t) = (p_2 \cdot P_g^2(t) + p_1 \cdot P_g(t) + p_0) \cdot e(t), \tag{3.21b}$$

$$\frac{d}{dt}E(t) \leq -P_b(t) - \beta(t) \cdot P_b^2(t), \tag{3.21c}$$

$$\beta(t) = \frac{\tilde{C} \cdot R}{2(E(t) + \Delta E)}, \tag{3.21d}$$

$$P_g(t) \in [0, P_{g,\max} \cdot u(t)], \tag{3.21e}$$

$$P_b(t) \in [P_{b,\min}(t), P_{b,\max}(t)]. \tag{3.21f}$$

3.2.3 Optimal Engine On/Off Strategy

In order to find the optimal engine on/off strategy, the optimal control problem of Section 3.1.3 needs to be solved with the model equations (3.21) instead of (3.17). The Hamiltonian function of the new optimal control problem [20] is

$$H(.) = P_f(t) - \lambda(t)\left(P_b(t) + \beta(t)P_b^2(t)\right). \tag{3.22}$$

By defining the equivalence factor as $s(t) = -\lambda(t)$ [21], the Hamiltonian function becomes

$$H(.) = P_f(t) + s(t)\left(P_b(t) + \beta(t)P_b^2(t)\right). \tag{3.23}$$

It is analyzed for two different cases [32]. In the first case, the engine-generator unit is considered to be on, while in the second case it is considered to be off:

$$H(.) = \begin{cases} H^{\text{on}}(.) & \text{given that } u(t) = 1 \\ H^{\text{off}}(.) & \text{given that } u(t) = 0, \end{cases} \tag{3.24}$$

where

$$\begin{aligned} H^{\text{on}}(.) &= (p_2 \cdot P_g^2(t) + p_1 \cdot P_g(t) + p_0) \\ &\quad + s(t)(P_b(t) + \beta(t)P_b^2(t)) && (3.25) \\ H^{\text{off}}(.) &= s(t)(P_b(t) + \beta(t)P_b^2(t)). && (3.26) \end{aligned}$$

In a first step, the power limits (3.21e)-(3.21f) are neglected. In the case where $e(t) = 1$, the subbranch of the Hamiltonian $H^{\text{on}}(.)$ is minimized by

$$P_b^{o,\text{on}}(t) = \frac{2p_2P_{\text{req}}(t) + p_1 - s^o(t)}{2(s^o(t)\beta^o(t) + p_2)}, \tag{3.27}$$

while in the case where $e(t) = 0$,

$$P_b^{o,\text{off}}(t) = P_{\text{req}}(t), \tag{3.28}$$

since $P_g^{o,\text{off}} = 0$ and (3.21a) has to be fulfilled[3].

Minimizing the Hamiltonian function reveals that switching the engine on is not optimal unless the following condition is satisfied

$$H^{\text{on}}(P_b^{o,\text{on}}) \leq H^{\text{off}}(P_b^{o,\text{off}}). \tag{3.29}$$

[3]The case where $P_b^{o,\text{off}}(t) > P_{\text{req}}(t)$ can be ruled out by arguing that dissipating energy in the braking resistor is not optimal unless the requested power is negative and the buffer is already fully charged.

By inserting (3.27) and (3.28) into (3.29) and solving for the requested power, these steps lead to the condition

$$P_{\text{req}}(t) \geq \frac{\frac{1}{2}(p_1 - s^o(t)) + \sqrt{p_0 \cdot (s^o(t)\beta^o(t) + p_2)}}{s^o(t)\beta^o(t)} = \Theta^o_{\text{lim}}(t). \tag{3.30}$$

A more detailed analysis (see appendix of publication iii) reveals that the engine-generator power limits (3.21e) do not influence this power threshold, while the buffer power limits (3.21f) do. The overall switching condition is thus

$$P^o_{\text{lim}}(t) = \begin{cases} P_{b,\max}(t) & \text{if } \Theta^o_{\text{lim}}(t) > P_{b,\max}(t) \\ P_{b,\min}(t) & \text{if } \Theta^o_{\text{lim}}(t) < P_{b,\min}(t) \\ \Theta^o_{\text{lim}}(t) & \text{otherwise.} \end{cases} \tag{3.31}$$

From (3.21d), the variable β is a function of the state variable $E(t)$, and thus the optimal power threshold is a function of the optimal equivalence factor $s^o(t)$ and the optimal state trajectory $E^o(t)$.

Overall, the optimal engine on/off strategy is given by

$$e^o(t) = \begin{cases} 1 & \text{if } P_{\text{req}}(t) \geq P^o_{\text{lim}}(t) \\ 0 & \text{else.} \end{cases} \tag{3.32}$$

Figure 3.5 illustrates P^o_{lim} as a function of the equivalence factor $s(t)$ and the buffer energy $E_b(t)$ in the case of a serial hybrid electric bus equipped with a 25 kWh battery. The threshold is calculated based on the original nonlinear open-circuit voltage curve data.

In the remainder of this article, the notation $e(P_{\text{req}}(t), E_b(t), s(t))$ is adopted to indicate the on/off strategy that corresponds to some given trajectories of $P_{\text{req}}(t)$, $E_b(t)$ and $s(t)$.

3.2.4 Necessary Conditions for Optimality

As illustrated by Figure 3.6, the findings of the previous section allow the engine on/off strategy to be calculated for given trajectories of the equivalence factor and the state variable $e_{\text{in}}(P_{\text{req,in}}, E_{b,\text{in}}, s_{\text{in}})$ (first block in Figure 3.6). Then, convex optimization can be applied to calculate the optimal power split for the predefined on/off strategy $e_{\text{in}}(.)$ (second block

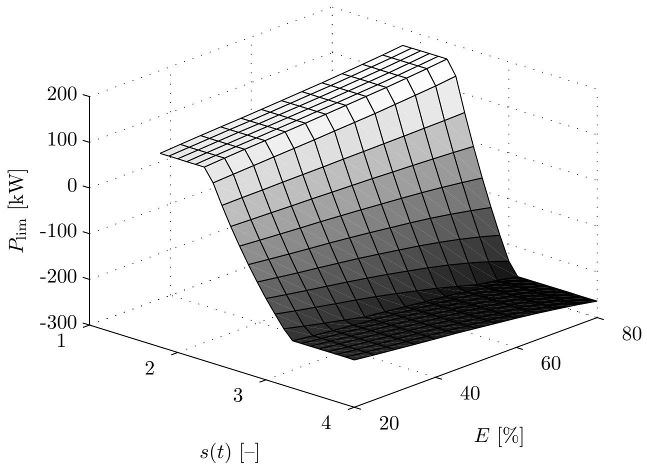


Figure 3.5: Engine on/off threshold in function of the equivalence factor and the state of charge. The map was evaluated using the original open-circuit voltage curve of a 25 kWh battery.

in Figure 3.6). As a result, the convex solver provides the optimal trajectories of the equivalence factor[4] $s_{\text{out}}(t)$ and the state variable $E_{b,\text{out}}(t)$ that correspond to the specific on/off strategy $e_{\text{in}}(t)$. Thus, the solution found by the convex solver is optimal for the entire energy management problem (3.16) if and only if the optimal engine on/off strategy $e_{\text{in}}(t) \equiv e^o(t)$ is used as an input[5], i.e.,

$$\text{if } e_{\text{in}}(t) \equiv e^o(t) \;\Rightarrow\; s_{\text{out}}(t) \equiv s^o(t), \quad \text{and } E_{b,\text{out}}(t) \equiv E_b^o(t). \tag{3.33}$$

Furthermore, if the on/off strategy is calculated based on the optimal trajectories $s^o(t)$ and $E^o(t)$, it will be the optimal on/off strategy, i.e.,

$$e(P_{\text{req}}, E^o(t), s^o(t)) \equiv u^o(t). \tag{3.34}$$

Combining (3.33) and (3.34) reveals that

$$\text{if } \; s_{\text{in}}(t) \equiv s^o(t), \quad \text{and} \quad E_{\text{in}}(t) \equiv E^o(t) \tag{3.35}$$

$$\Rightarrow s_{\text{in}}(t) \equiv s^o(t), \quad \text{and} \quad E_{\text{in}}(t) \equiv E^o(t), \tag{3.36}$$

[4]The convex solver provides the Lagrangian multiplier in the form of a dual variable that allows the calculation of the equivalence factor.

[5]The equivalence sign '≡' is used to indicate that $e_{\text{in}}(t) = e^o(t), \forall t \in [0, t_f]$.

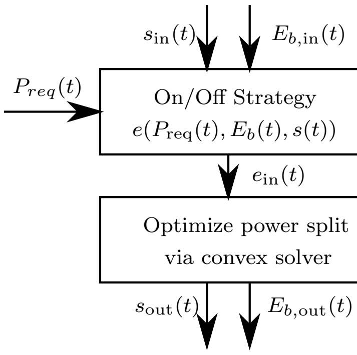


Figure 3.6: Sequential optimization of the engine on/off strategy and the power split.

and therefore, every globally optimal solution satisfies

$$s_{\text{out}}(t) \equiv s_{\text{in}}(t), \quad \text{and} \quad E_{\text{out}}(t) \equiv E_{\text{out}}(t). \tag{3.37}$$

Thus (3.37) is a *necessary* condition for a globally optimal solution.

In order to make sure that a solution satisfying the necessary contdition (3.37) is truly the globally optimal solutoin, one would need to prove that there exists no other solution that satisfies (3.37) and results in a lower total cost. This proof is generally not straightforward and not investigated further. In the case studies described in the following sections, it will be shown that our proposed method indeed yields the same results as the dynamic programming algorithm.

3.2.5 Solution Finding Procedure

Figure 3.7 illustrates an iterative algorithm that can be used to find the optimal trajectory of the equivalence factor and thus the optimal solution to the energy management problem. Accordingly, the equivalence factor and the state variable are initialized by some constant values $s_{\text{in}}^{j=0}(t)$ and $E_{b,\text{in}}^{j=0}(t)$. Then, the corresponding on/off strategy $e_{\text{in}}(t)$ is computed, which is then used to optimize the power split by solving the convex problem (3.17). To converge towards the optimal equivalence factor $s^o(t)$, the

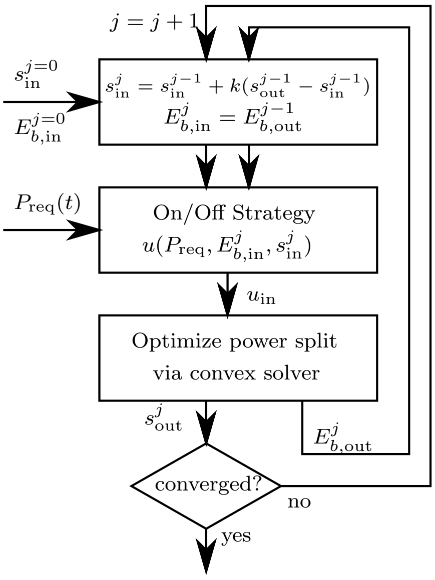


Figure 3.7: Iterative search algorithm to find the optimal trajectories of the equivalence factor and the state variable.

following update law is proposed (with j denoting the iteration index):

$$s_{\text{in}}^{j+1}(t) = s_{\text{in}}^{j}(t) + \kappa \cdot (s_{\text{out}}^{j}(t) - s_{\text{in}}^{j}(t)), \tag{3.38}$$

$$E_{b,\text{in}}^{j}(t) = E_{b,\text{out}}^{j}(t). \tag{3.39}$$

As shown in Fig. 3.7, the entire sequence is repeated until a given convergence criterion is fulfilled, e.g. until the root mean square error

$$\epsilon^{j} = \sqrt{\frac{1}{t_f} \int_0^{t_f} \left(s_{\text{out}}^{j}(t) - s_{\text{in}}^{j}(t) \right)^2 dt} \tag{3.40}$$

is smaller than a predefined tolerance.

The equivalence factor is initialized with a value that is large enough to ensure that the resulting initial on/off strategy is feasible and that the problem is solvable by the convex solver. If $s_{\text{in}}^{j}(t)$ was initialized with a value larger than $s^{o}(t)$, the resulting total engine-on time would be increased compared to the optimal case, giving more time to recharge the

Table 3.3: Battery Model Parameters

Parameter	Symbol	Value	Unit
Energy Capacity	E_0	$s_b \in$[10,30]	[kWh]
Nominal Power	P_{nom}	$6.46 \cdot s_b$	[kW]
Equivalent Capacity	$\tilde{C}$	$30.9/s_b$	[kF]
Internal Resistance	R	$2.62 \cdot s_b$	[mΩ]
Energy Offset	ΔE	$0.74 \cdot s_b$	[kWh]
Max. Power Parameter	$b_{u,1}$	$2.00 \cdot s_b$	[kW/kWh]
Max. Power Parameter	$b_{u,0}$	$4.46 \cdot s_b$	[kW]
Mass	m_b	$27.5 \cdot s_b$	[kg]

buffer. Consequently, the resulting equivalence factor $s^j_{\text{out}}(t)$ would be lower than $s^o(t)$. A similar argument holds for the case where $s^j_{\text{in}}(t) < s^o(t)$ and thus the optimal value of the equivalence factor lies within $s^o(t) \in [s^j_{\text{in}}(t), s^j_{\text{out}}(t)]$. Therefore, practical values of the feedback gain are found to be around $\kappa = 0.5$.

3.3 Case Studies

3.3.1 Hybrid Electric Bus with a Battery

First, this section presents a small example to illustrate how the algorithm described above works. Then, a sizing case study is presented where a battery pack is optimized for a serial hybrid electric bus for four different bus lines individually. The parameters used for the convex battery model are listed in Table 3.3 as a function of the energy capacity. The problem was implemented in Matlab [34]. The convex problem was parsed with CVX [35] and solved with SeDuMi [36]. The dynamic programming solution was found using the dpm function [19]. Note that the charge conservation condition $E(t_f) = E(0)$ was relaxed to $E(t_f) \geq E(0)$. This relaxation has been shown to actually improve numerical precision of the algorithm [4].

Driving Cycles

Figure 3.8 shows the velocity and altitude profile of the four bus lines considered in the case study. The auxiliary power demand and the number of passengers were recorded together with the data shown along the driving

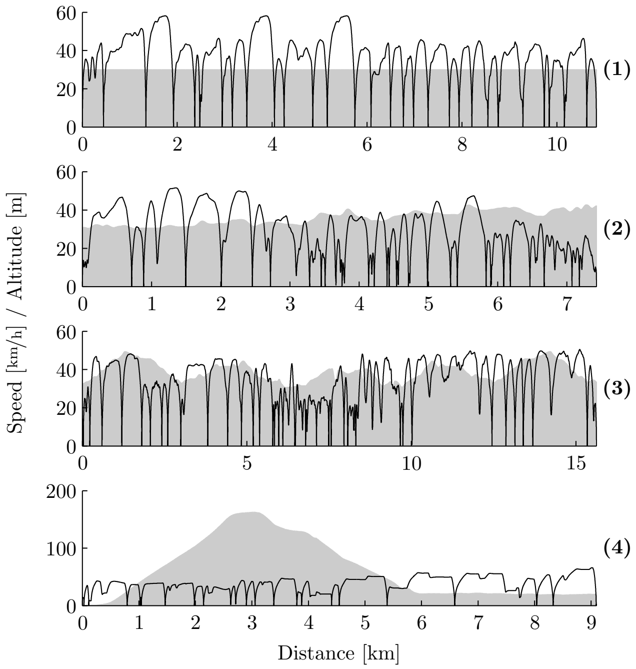


Figure 3.8: Speed (black lines) and altitude profiles (gray areas) of the bus lines 1 through 4.

missions. All the data were subsequently resampled at a rate of 1 Hz.

Lines 1-2 are rather flat inner city lines recorded in the cities of Braunschweig and Düsseldorf, while Line 3 is a slightly more hilly bus line measured in Hamburg. Line 4 is an artificial cycle that was used to validate the drivetrain of the bus on a testbench. This cycle includes a severe altitude profile and prolonged phases of driving at the power limits of the bus.

Example Solution

The proposed algorithm is now applied to a 16 kWh battery hybrid electric bus driving on Line 1. The upper plot of Figure 3.9 shows the evolution of the state variable of both the solution found by the proposed algorithm (black line) and the optimal solution obtained using dynamic programming (thick gray line). The lower plot shows the evolution of the equivalence

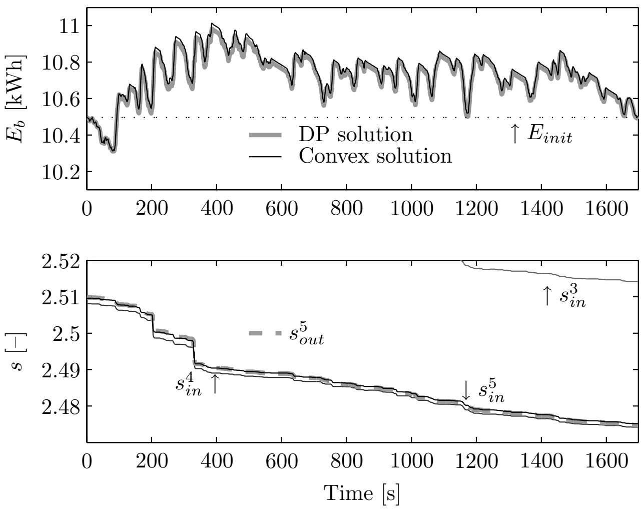


Figure 3.9: Iterations 3, 4 and 5 when applying the proposed algorithm to a 16 kWh battery hybrid electric bus on Line 1. The globally optimal solution found by dynamic programming is indicated by a thick gray line in the upper plot.

factor s^j_{in} for the iterations 3, 4 and 5 (differently shaded solid lines) and the final equivalence factor s^5_{out}. Clearly, the algorithm converges towards a solution very similar to the one found by dynamic programming.

In the proposed algorithm, the equivalence factor was initialized with a constant value of $s^{\text{in},k=0}(t) = 3$, while the buffer energy was initialized by $E_b^{\text{in},k=0}(t) = E_{\text{b,init}}\ \forall\ t \in [0, t_f]$. The gain was chosen $\kappa = 0.5$. After five iterations the root mean square error ϵ^j was smaller than $1 \cdot 10^{-3}$ and the algorithm terminated. The fuel consumption found is 35.26 l/100km, which is 0.02 % lower than the one found by dynamic programming (35.27 l/100km). The solution was found in less than 35 s, while the dynamic programming approach required 417 s.

The equivalence factor is not constant because the battery open circuit voltage is an affine function of the battery charge. Furthermore, due to discretization, dynamic programming is not completely free from numerical errors, which is the reason for the fact that the solution found is slightly worse than the one found by convex optimization. However,

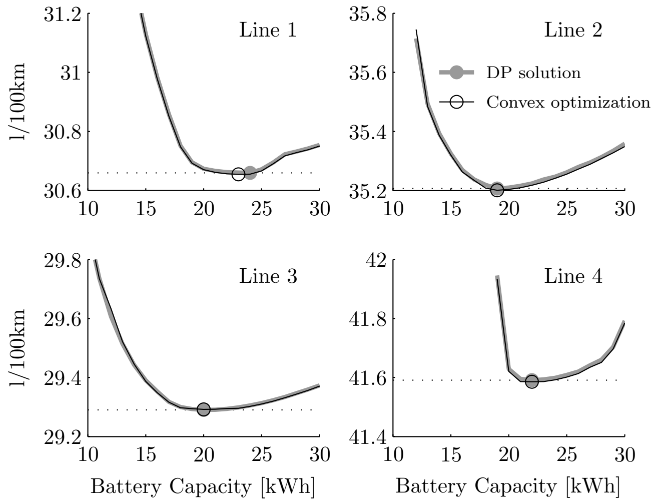


Figure 3.10: Fuel consumption as a function of the battery capacity, normalized by the best possible fuel consumption on each line. The optimal battery size found is indicated by a grey dot and a circle, respectively.

the proposed algorithm yields a more precise solution in less time than dynamic programming.

Sizing Study

This subsection presents a case study in which the fuel-optimal battery size of the serial hybrid bus is sought for each bus line individually. To do so, the energy management problem was solved with the proposed method for battery capacities between 10 and 30 kWh in steps of 1 kWh. Figure 3.10 shows the optimal fuel consumption and the corresponding battery capacity. The results from the iterative algorithm are indicated by a thin black line. The identical problem was solved using dynamic programming as well, providing the globally optimal solution indicated by a gray thick line. The fuel optimal battery size for each line is indicated by a black circle and a gray disk for the iterative and the dynamic programming solution, respectively. For battery sizes smaller than the optimal one, the fuel consumption increases because the battery cannot absorb all the energy from recuperation, while for an increased battery size, due to

Table 3.4: Model Parameters of the Supercapacitors

Parameter	**Symbol**	**Value**	**Unit**
Energy Capacity	E_{nom}	$s_c \in$[300,2000]	[Wh]
Nominal Power	P_{nom}	$0.42 \cdot s_c$	[kW]
Equivalent Capacity	C	$7.81/s_c$	[kF]
Internal Resistance	R	$0.22 \cdot s_c$	[mΩ]
Max. Power Parameter	b_1	$1.04 \cdot s_c$	[W/Wh]
Max. Power Parameter	b_0	$0.12 \cdot s_c$	[kW]
Mass	m_b	$0.4 \cdot s_c$	[kg]

the additional weight of the vehicle, fuel consumption increases as well. The results from the iterative algorithm match the results from dynamic programming with an error of less than 0.09 % in fuel consumption. By choosing the convergence criterion to be the root mean square of the error ϵ^j being smaller than $1 \cdot 10^{-3}$, the iterative algorithm terminated with 6.7 iterations on average. The maximum error regarding the fuel-optimal battery size is 1 kWh on Line 1. This error is likely to be reduced if a finer grid of battery capacity is chosen. The computation time of both the proposed and the dynamic programming algorithms were measured on a personal computer with an Intel Core i7 CPU 2.80 GHz 64-bit processor. The dynamic programming algorithm required an average computation time of more than 489 s per battery size and bus line combination. By contrast, the proposed algorithm required an average computation time of 61 s.

3.3.2 Hybrid Electric Bus with Supercapacitors

The previous section demonstrated that the proposed algorithm exhibits good convergence properties in the case of a battery hybrid electric bus. However, with such large battery energy buffers, the state constraints, i.e. the upper and lower bound of the battery energy content, rarely become activated. In order to investigate whether the algorithm still shows good convergence properties in the presence of active state constraints, it is now being applied to a serial hybrid electric bus with supercapacitors. The buffer parameters as a function of the energy capacity for this study are listed in Table 3.4.

Example Solution

Figure 3.11 shows iterations 2, 3, 5 and 8 when the proposed algorithm is applied to a 400 Wh serial hybrid electric bus driving on Line 1. The upper graph shows the evolution of the state trajectory, while the lower graph shows the equivalence factor that was used to calculate the on/off strategy. Clearly, the upper state constraint is being activated regularly, resulting in a well observable jump to a lower value in the equivalence factor. Furthermore, at time instances 84 s, 198 s and 1174 s, the buffer needs to be fully discharged in order to maximize recuperation in the braking phase that follows. During those braking intervals, the equivalence factor jumps to a value of zero, because all energy that is left in the buffer before entering the braking phase has to be wasted. Nevertheless, the algorithm converges towards the solution calculated with dynamic programming. After 8 iterations, the root mean square error between ϵ^j was smaller than $1 \cdot 10^{-2}$ and the algorithm terminated. The fuel consumption found by the proposed algorithm is 35.73 l/100km, while the one found by dynamic programming is 35.75 l/100km.

Sizing Study

Once again, a sizing study is performed to find the optimal supercapacitor capacity of a hybrid electric bus on the four bus lines shown in Sect. 3.3.1. The capacity is varied from 300 to 2000 Wh in 100 Wh steps. Figure 3.12 shows the fuel consumption as a function of the supercapacitor size for each of the bus lines. For lines 1-3, there exists an optimal capacity within the range studied. A lower capacity leads to an increased use of the braking resistor, while a higher capacity leads to a heavier vehicle and thus an increased fuel consumption. On Line 4, which contains a long downhill section, the optimal buffer capacity is larger than 2000 Wh. With a capacity below 400 Wh, Line 4 is not driveable because the supercapacitor does not provide enough power. Obviously, the proposed algorithm delivers the same results as the dynamic programming algorithm. The maximum deviation in fuel consumption between the two solutions is 0.2 %. The maximum error of the optimal buffer design found is 100 Wh, which again is likely to be reduced if a finer grid of capacity is chosen. The iterative algorithm terminated after 7.6 iterations on average with a mean evaluation time of 94 s, while the dynamic programming algorithm required 478 s on average.

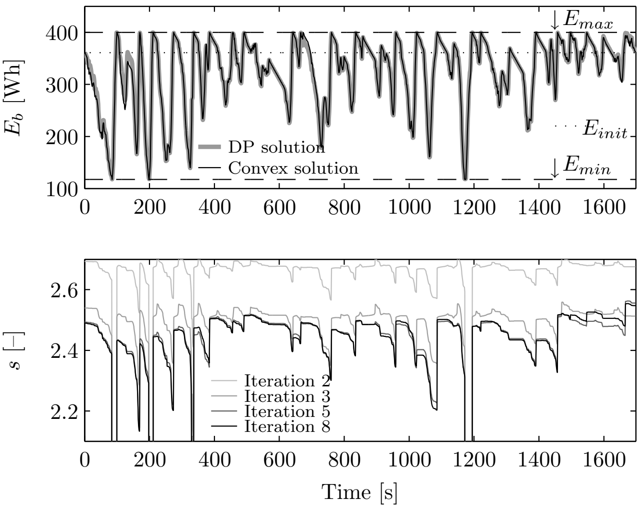


Figure 3.11: Iterations 2,3,5 and 8 the proposed algorithm is applied to a 400 Wh supercapacitor hybrid electric bus on Line 1. The globally optimal solution found by dynamic programming is indicated by a thick gray line in the upper plot.

Comparison between Battery and Supercapacitor Designs

A comparison of the optimal designs found by the proposed and the dynamic programming algorithms is shown in Table 3.5 for both the battery and the supercapacitor hybrid electric bus. For an optimally designed bus, the fuel consumption is very similar for both types of buffers. On Lines 1-3, the supercapacitors have a slight advantage (up to 1.5 % reduction of fuel consumption) due to the fact that the optimal design weighs less than that in the case of batteries; however in practice this difference seems negligible. In the case of a hilly cycle such as that of Line 4, the supercapacitor would have to be severely oversized in order to be able to capture all energy from recuperation.

Overall, from the viewpoint of energy efficiency, supercapacitors are an alternative to batteries on rather flat inner city cycles. On hilly cycles where a lot of energy has to be recuperated, batteries are better suited.

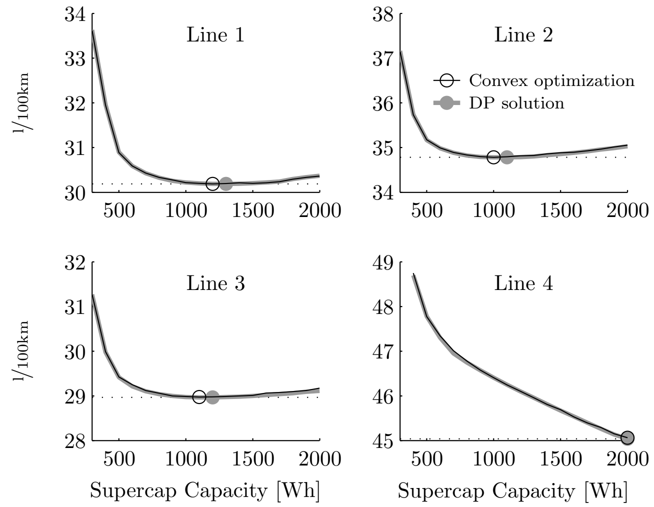


Figure 3.12: Fuel consumption as a function of the supercapacitor capacity, normalized with respect to the fuel-optimal solution from dynamic programming. The optimal capacity found is indicated by a grey dot and a circle, respectively.

3.4 Conclusion

This chapter addresses an open issue that usually occurs when attempting to find the globally optimal solution of the energy management problem of a hybrid electric vehicle using convex optimization. The open issue refers to binary or integer decision variables, such as the engine on/off decision, which cannot be included in the problem formulation because of their non-convexity. Up to now, researchers have used suboptimal control strategies to define these decisions prior to solving the actual energy management problem. In this paper, we analytically analyze the problem in the case of a serial hybrid electric bus and derive the engine on/off strategy to be a function of the drive cycle data, the state variable and the equivalence factor. These findings are then used to derive an iterative algorithm that converges to the globally optimal solution after a few iterations. The algorithm is shown to deliver optimal results in less time than dynamic programming even in the presence of active state constraints.

Future work is to include testing the algorithm in the case where the buffer capacity is included as a sizing variable in the convex program. Fur-

Table 3.5: Optimal Designs found by the proposed algorithm

	Bus Line	**1**	**2**	**3**	**4**
Battery	$E_{\mathbf{nom}}$ [kWh]	23	19	20	22
	FC [l/100km]	30.66	35.21	29.29	41.59
	$E_{\mathbf{nom}}^{DP}$ [kWh]	24	19	20	22
	FC^{DP} [l/100km]	30.66	35.20	29.29	41.59
Supercapacitor	$E_{\mathbf{nom}}$ [Wh]	1200	1000	1100	>2000
	FC [l/100km]	30.19	34.78	28.98	<45.06
	$E_{\mathbf{nom}}^{DP}$ [Wh]	1300	1100	1200	>2000
	FC^{DP} [l/100km]	30.20	34.79	28.98	<45.04

thermore, the method should be extended to other hybrid topologies, such as the parallel topology, or hybrid powertrains with two buffer systems. Finally, the algorithm should be extended to become capable of handling integer decision variables, such as gear shifts.

Part II

Online Energy Management

Chapter 4

Stochastic Dynamic Programming

Hybrid electric vehicles require an energy management system to define the power split, i.e., to determine how the power that is requested by the driver is split up between the electric energy buffer and the combustion engine. A number of energy management strategies have been proposed in the literature. Early heuristic strategies are based on fuzzy logic [37, 38] or on rules obtained using engineering intuition [39, 40, 41]. Unfortunately, those types of controllers are difficult to tune and fail to guarantee optimal performance. Better strategies are based on optimal control theory, such as the well known equivalent consumption minimization strategy [42, 43, 44], which delivers good results when applied to battery hybrid electric vehicles. Unfortunately, this strategy neglects the state constraints and requires the equivalence factor to be approximated as a constant [21]. In the case of the serial hybrid electric bus presented in Chapter 1 these assumptions are invalid because supercapacitors with a rather limited storage capacity are used as an electric buffer.[1]

Stochastic dynamic programming [2, 3] has been proposed by many researchers for solving the energy management problem, and extensive simulation results have been presented [45, 46, 47, 48, 49, 50]. Considering a hybrid electric bus equipped with supercapacitors, the application of stochastic dynamic programming is supported by two facts: First, in the context of public transportation, the driver behavior is very repetitive and may be approximated by a Markov chain [3] reasonably well. Second, the stochastic dynamic programming algorithm neither neglects the state constraints, nor requires any assumptions regarding the buffer parameters.

[1] With supercapacitors, the open-circuit voltage is a linear function of the actual charge $U_{sc} = \frac{Q(t)}{C}$. Applying Pontryagins minimum principle reveals that the equivalence factor may not be approximated as a constant in this case.

The advantage of such a controller is that i) optimal performance is guaranteed with respect to the available information, ii) the controller can be stored in a look-up table and can therefore easily be evaluated using limited computational resources, and iii) controller synthesis is a systematic procedure that can easily be extended in a way that the resulting controller optimally trades off multiple objectives, e.g., fuel economy vs. driveability, while at the same time it monitors additional system variables such as engine temperature, etc. Additionally, the concept offers the possibility to customize energy management controllers for specific bus lines.

The drawbacks of this concept are i) the relatively high computational burden for controller synthesis, and ii) the fact that it is not easily feasible to integrate online predictive information about the bus line into the offline controller synthesis process, e.g., the elevation profile of the bus line. The former drawback can be overcome by an efficient implementation which will be described below, while the latter can be handled by a predictive extension as discussed in Chapter 5.

The algorithm presented in this chapter is an extension to the basic stochastic dynamic programming algorithm. Instead of expressing the full model as a Markov chain, the state update function is split up in two parts: The driver behavior is expressed as a stationary Markov chain, while the vehicle is modeled using deterministic equations. A similar decomposition has been proposed in [50].

In this chapter the performance of the controller is tested in simulation, as well as on the real vehicle using a standardized test procedure that allows comparing fuel consumption results to those of other vehicles. Simulation results show that the performance of the controller is close to optimal, with an increase of only 3 to 5% with respect to the global optimal solution obtained with dynamic programming.

This chapter is based in part on the work of Martin Widmer [51].

4.1 Problem Formulation

The controller sought in this chapter is a real-time capable, causal controller that does not rely on any information about the future driving profile. The only information required consists of the stochastic properties of the vehicle speed and the electric traction power as well as of the corresponding real time measurement data. The vehicle model used for controller synthesis is the one presented in Chapter 1, augmented by one additional state variable representing the engine temperature. The temperature model is required to prevent the controller of permitting engine stops under cold conditions.

This section first introduces some minor adaptations on the vehicle model presented in Chapter 1, followed by the formulation of the optimal control problem.

4.1.1 Vehicle Model

The vehicle model used in this chapter has three state variables

$$x_k = \begin{pmatrix} E_{b,k} \\ \omega_{g,k} \\ \vartheta_{e,k} \end{pmatrix}, \tag{4.1}$$

where $E_{b,k}$ is the energy currently present on the supercapacitor, $\omega_{g,k}$ is the rotational speed of the generator and $\vartheta_{e,k}$ is the engine coolant temperature.

Engine Generator Unit

The models presented in Chapters 1 and 3 assumed that the energy management unit controls the buffer power. Here, a different definition of the control input is used: The control input decides which out of several discrete operating points $u \in \{1, 2, \dots Q\}$ the engine generator unit is to be operated on. Those operating points coincide with the curve of optimal efficiency introduced in Section 1.5, Figure 1.5. The operating points in the presented implementation correspond to 0, 20, 50, 60, 70, 80, 90, 120 and 165 kW of electric output power. In the case of zero output power the engine can be either off or idling. Each operating point is defined by a certain rotational speed

$$\omega_{g,k} \in \left\{\omega^1, \dots, \omega^Q\right\}, \tag{4.2}$$

and a certain electric output power

$$P_{g,k} = \begin{cases} P_g^u & \text{if } \omega_k = \omega^u \leq \omega_{k-1}, \\ 0 & \text{else,} \end{cases} \tag{4.3}$$

where $P_g^u \in \{P_g^1, \cdots, P_g^Q\}$. The second case of (4.3) reflects the fact that the generator is not allowed to brake the engine during an acceleration, in order to achieve a quick transition between the operating points. Furthermore, the generator is not allowed to motor the engine except during the start-up phase.

The implementation of such discretized controls compromises optimality to a small extent with respect to a continuous control input. However, a comparison of the two variants using deterministic dynamic programming reveals that the deterioration is negligible since the engine is operated at maximum efficiency most of the time. Furthermore, the discrete definition results in a piecewise constant reference value for the EGU speed controller, thus avoiding small variations that would occur otherwise.

Furthermore, the fuel consumption that corresponds to each discrete operating point and to all possible transitions between any two operating points has been identified from a dynamic model of the engine generator unit involving the efficiency maps of the engine and the generator. The values have been stored in a two-dimensional look-up table

$$\dot{m}_{f,k} = f_{\text{fuel}}(x_k, u_k). \tag{4.4}$$

This model assumes that the transition between any two operating points can be realized within one time step. With a temporal discretization of one second and the distribution of operating points chosen in the actual vehicle, this assumption corresponds well with reality for all operating points up to 90 kW. For operating points with a power above that value, the turbocharger dynamics usually induce a longer transition time. In these cases, the fuel map and the state update function are adjusted to compensate for the errors of this assumption.

Engine Temperature

The energy management has the ability to stop and restart the engine whenever adequate. However, if the engine is cold, e.g., shortly after the vehicle is put into operation, stopping the engine should be avoided in order to reduce thermal stress and keep emissions low. To find a controller that monitors the engine temperature, the model used for controller synthesis is augmented by the engine coolant temperature as an additional state variable.

Table 4.1: Parameters of the engine temperature model.

Parameter	Symbol	Value	Unit
thermal capacity	c_e	480	[J/kg]
convective term	κ_e	200	[W/K]
ambient temperature	ϑ_{amb}	21	[deg C]

The dynamics of the engine coolant temperature ϑ_e are modeled by a basic zero-dimensional, lumped parameter thermodynamic model, where the engine is considered to be a metal block with equally distributed temperature, discretized using the Euler forward method

$$\begin{aligned}\vartheta_{e,k+1} &= \vartheta_{e,k} + \frac{1}{c_e}\left(\dot{q}_{\mathrm{in},k} - \dot{q}_{\mathrm{out},k}\right) \cdot \Delta t \qquad (4.5)\\ &= \vartheta_{e,k} + \frac{1}{c_e}\left(\frac{1}{2}\big(\dot{m}_{f,k} H_{\mathrm{lhv}} - T_{e,k}\omega_{e,k}\big) - \kappa_e(\vartheta_{e,k} - \vartheta_{\mathrm{amb}})\right) \cdot \Delta t.\end{aligned}$$

In this equation the following assumption is made: Half of the fuel energy that is not transformed into mechanical energy is transferred to the cooling system, while the other half is transported to the ambient together with the exhaust gas. The engine coolant itself is cooled via convection. The parameters are listed in Table 4.1. A rudimentary validation of this model has been conducted by comparing the time constants of the model and the real system during operation.

4.1.2 Markov Chain

The basic idea behind stochastic dynamic programming is that the driver behavior can be modeled and predicted by a stationary Markov chain. This section illustrates the concept.

The vehicle speed and the electric traction power are considered disturbances to the energy management in the sense that they are measurable online but are not known in advance. Consider the disturbances as a vector

$$w_k = \begin{pmatrix} v_k \\ P_{\mathrm{req},k} \end{pmatrix} \in W = \{w^1, w^2, \dots, w^R\}, \qquad (4.6)$$

where v_k and $P_{\mathrm{req},k}$ are the vehicle speed and the electric traction power at timestep with index k. Furthermore, consider that the disturbance space W has been discretized using R discrete values.

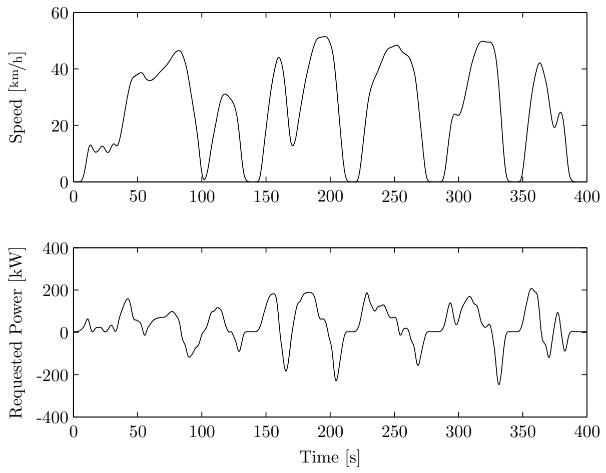


Figure 4.1: Speed and requested power along a driving mission. Figure 4.2 shows the same data on the plane representing speed vs. requested power.

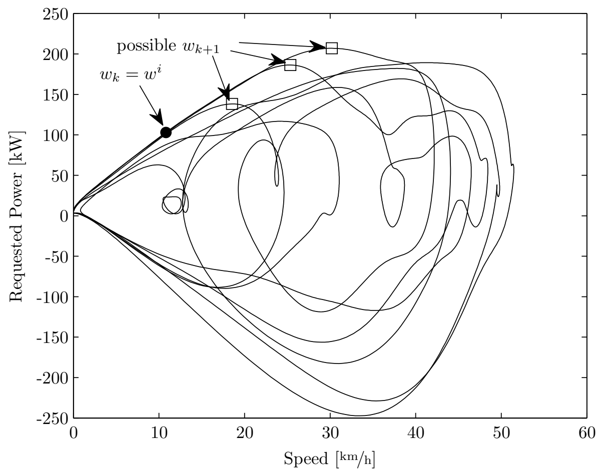


Figure 4.2: Concept of a Markov chain illustrated in the plane that represents speed vs. traction power: If the disturbance at timestep k is $w_k = w^i \in W$, then there exists a two-dimensional probability density function that describes the probability that $w_{k+1} = w^j$ at timestep $k+1$ for all $w^j \in W$.

Figures 4.1 and 4.2 illustrate the process of generating a Markov chain. Figure 4.1 shows the velocity profile along a driving mission and the corresponding trajectory of the requested power. If the same data is plotted in the speed-power plane as in Figure 4.2, it becomes clear that the driver behavior can be predicted to some extent. Consider that the disturbance is measured to be $w_k = w^i$ at timestep k indicated by a dot in Figure 4.2. Despite the fact that there are several possibilities for what happens next (possible w_{k+1} indicated by boxes), it is obvious that the driver is in acceleration mode. Thus, given the measurement $w_k = w^i$, there is a high probability that the driver will continue to accelerate the vehicle and that the velocity and the requested power will continue to rise during the upcoming timesteps. Contrarily, there is a low probability for a negative power request during the next timestep.

By examining the trajectories of the velocity and the requested power for several complete missions in the same way as above, a transition probability matrix can be obtained for each velocity-power pair in the set W

$$\begin{aligned} p_{ij} &= P(w_{k+1} = w^j \mid w_k = w^i) \\ &= \frac{\#(w_{k+1} = w^j \mid w_k = w^i)}{\#(w_k = w^i)} \quad \forall\, w^i, w^j \in W, \end{aligned} \tag{4.7}$$

where each entry p_{ij} denotes the transition probability from w_i to w_j, which is determined by counting the number of times w^j is visited exactly one timestep after w^i was visited, divided by the number of times w^i was visited in total.

Storing these transition probabilities for all possible permutations of j and i in a matrix yields the Markov chain p_{ij}. Each row of the matrix (i fixed) represents a discrete two-dimensional probability density function for the transition starting from $\omega_k = \omega^i$, and thus $\sum_{j=1}^{R} p_{ij} = 1$ with $p_{ij} \geq 0$.

In the following, the variable w denotes a deterministic measurement of the disturbance vector, while $\bar{w}$ denotes a stochastic variable with the probability distribution

$$\bar{w} \sim \mathcal{P}(w). \tag{4.8}$$

The probability distribution function $\mathcal{P}$ is described by the Markov chain p_{ij}.

Implementation Issues

In order to avoid numerical problems, special care has to be taken when calculating the matrix p_{ij}. First, given a measured value of ω_k, which is unlikely to coincide perfectly with one of the discrete values in W, linear interpolation is used to evaluate the transition probability starting from ω_k. Second, certain combinations of vehicle speed and electric power $w^i \in W$ may not be visited in the data set that is used to generate the Markov chain and thus, the corresponding transition probability density function cannot be identified, since division by zero occurs in (4.7). Therefore, each measurement of w is overlaid with a normally distributed measurement uncertainty, and is thus spread over several discrete points within W. Such a spread ensures that there is a valid transition probability density function for all discrete points contained in W.

4.1.3 Optimal Control Problem

A basic assumption required for solving the energy management problem using a stochastic dynamic programming algorithm is that the stochastic properties of all disturbances, i.e., vehicle speed and traction power, are known and can be expressed as a Markov chain. In order to obtain a stationary control policy, i.e., a control policy that is independent of time, the problem is regarded as an infinite horizon problem, i.e., the bus never finishes its mission. Alternatively, the problem could be formulated as a discounted problem or a problem including a terminal state [3], however these approaches require additional parameters to tune.

The optimal control problem is summarized as follows: find an admissible, stationary control policy $u_k = \mu^o(x_k, w_k)$ that minimizes the cost functional (4.9) and satisfies the constraints (4.10–4.13)

$$\min_{\mu(.)} \lim_{N\to\infty} \sum_{k=0}^{N} \underset{w_k}{E}\left\{g(x_k, \mu(.), w_k)\right\} \tag{4.9}$$

$$\begin{aligned} x_{k+1} &= f(x_k, \mu(.), w_k) && (4.10)\\ x_k &\in X && (4.11)\\ u_k &\in U && (4.12)\\ w_k &\sim \mathcal{P}(w_{k-1}) && (4.13)\end{aligned}$$

$$\text{for all } k = 0, 1, \ldots, \infty.$$

Since the disturbance w_k is a stochastic variable, the cost functional (4.9) includes the expected value of the stage cost $g(x_k, u_k, w_k)$. Apart from the fuel energy, the stage cost is allowed to contain several other weighted penalty factors to trim the controller towards a behavior that is acceptable in a vehicle

$$g(x_k, u_k, w_k) = H_{\mathrm{lhv}} \cdot \dot{m}_{f,k} + g_1(x_k, u_k, w_k) + g_2 \dots \tag{4.14}$$

4.2 Stochastic Dynamic Programming

The basic formulation of stochastic dynamic programming assumes that the full process model (deterministic or stochastic) is a Markov chain. In the slightly different derivation described in this chapter, the state update function is assumed to be comprised of both a stochastic model of the driver behavior expressed as a Markov chain, as well as a deterministic vehicle model.

Due to the discrete nature of dynamic programming, the time, the state, the control and the disturbance spaces must be discretized. The state space is represented by the set $X = \{x^1, x^2, \dots, x^P\}$. The control and the disturbance spaces have already been discretized to $W = \{w^1, w^2, \dots, w^R\}$ and $U = \{1, 2, \dots, Q\}$. The remainder of this chapter assumes that the variables x, u and w are always members of these discrete sets.

In an infinite horizon problem, starting from any given combination of a state x^p and a disturbance w^i at any given time with index k, the cost to finish the problem is always the same, i.e., the optimal cost-to-go $\mathcal{J}^o$ is not a function of the time, and hence it is stationary

$$\mathcal{J}_k^o(x, w) = \mathcal{J}_{k+1}^o(x, w) = \mathcal{J}^o(x, w), \tag{4.15}$$

where $p = 1, \dots, P$ and $i = 1, \dots, R$. Therefore, the regular dynamic programming iteration

$$\mathcal{J}_k^o(x, w) = \min_{\mu(.)} \left[g(x, \mu(.), w) + \underset{\bar{w} \sim \mathcal{P}(w)}{E} \left\{ \mathcal{J}_{k+1}^o(f(x, \mu(.), w), \bar{w}) \right\} \right], \tag{4.16}$$

can be reformulated into an implicit equation

$$\mathcal{J}^o(x, w) = \min_{\mu(.)} \left[g(x, \mu(.), w) + \underset{\bar{w} \sim \mathcal{P}(w)}{E} \left\{ \mathcal{J}^o\left(f(x, \mu(.), w), \bar{w}\right) \right\} \right]. \tag{4.17}$$

If the optimal cost-to-go function is known, the optimal policy is given by the argument minimizing the right-hand side of (4.17) . The expected value E of the cost function is evaluated via a summation over all possible future values of the disturbance vector $\bar{w}$ and their corresponding probabilities stored in the Markov chain matrix p_{ij}.

4.2.1 Value Iteration

The simplest way of solving (4.17) is called value iteration [3]. First, assume that a sub-optimal initial guess of the cost-to-go function $\mathcal{J}$ is available. Then let $T(.)$ be an operator that applies the following operation on $\mathcal{J}$

$$T(\mathcal{J}) = \min_{\mu(.)} \left[g(x, \mu(.), w) + \underset{\bar{w}\sim\mathcal{P}(w)}{E} \left\{ \mathcal{J}\left(f(x, \mu(.), w), \bar{w} \right) \right\} \right]. \tag{4.18}$$

Furthermore, let the repeated application of $T(.)$ on $\mathcal{J}$ be denoted as

$$T^{\alpha}(\mathcal{J}) = T(T^{\alpha-1}(\mathcal{J})), \tag{4.19}$$

where the superscript α denotes the number of iterations. With these definitions, the repeated application of $T(.)$ on $\mathcal{J}$ always converges to the optimal cost-to-go function, regardless of the initial guess of $\mathcal{J}$ [3], i.e.,

$$\lim_{\alpha\to\infty} T^{\alpha}(\mathcal{J}) = \mathcal{J}^{o}. \tag{4.20}$$

4.2.2 Modified Policy Iteration

Another algorithm is called policy iteration [3]. First, assume that an initial guess of the control strategy $\mu(x, w)$ exists. Then let $T_{\mu}(.)$ be an operator that applies the following operation on $\mathcal{J}$

$$T_{\mu}(\mathcal{J}) = g(x, \mu(.), w) + \underset{\bar{w}\sim\mathcal{P}(w)}{E} \left\{ \mathcal{J}\left(f(x, \mu(.), w), \bar{w} \right) \right\}. \tag{4.21}$$

The repeated application of $T_{\mu}(.)$ on $\mathcal{J}$ always converges to the true cost-to-go of this specific policy [3], i.e.,

$$\lim_{\alpha\to\infty} T_{\mu}^{\alpha}(\mathcal{J}) = \mathcal{J}_{\mu}, \tag{4.22}$$

which is called policy evaluation.

Obviously, policy iteration alone does not lead to the globally optimal solution. In order to find the true optimum, policy iteration has to be combined with value iteration. The algorithm takes the following form.

1. Initialization: Guess an initial policy $\mu^{\alpha=0}$.

2. Policy Evaluation: Given the current policy μ^{α}, evaluate $T_{\mu}(\mathcal{J})$ m times, i.e., $\mathcal{J}_{\mu^{\alpha}} = T^{m}_{\mu^{\alpha}}(\mathcal{J})$, as in (4.21).

3. Policy Improvement: Update the policy $\mu^{\alpha+1}$ as the minimizing argument of $T(\mathcal{J}_{\mu^{\alpha}})$, as in (4.18) .

4. Iteration: Iterate steps **2.** to **3.** until a stationary cost is found, i.e., until the condition $\mathcal{J}_{\mu^{\alpha}} = T(\mathcal{J}_{\mu^{\alpha}})$ is satisfied.

This algorithm usually converges much faster than value iteration. The parameter m is problem specific and has to be defined by the user. It determines the trade-off between the number of calculations needed per iteration and the rate of convergence of the algorithm.

4.2.3 Implementation Issues

A number of problems have to be considered when implementing this algorithm. These are listed below.

Gauss-Seidel Iteration

During step **2.**, the operation $T_{\mu}(\mathcal{J}_{\mu^{\alpha}})$ has to be evaluated m times. The mathematically correct implementation would be to store an intermediate cost-to-go function $\mathcal{J}^{i}_{\mu^{\alpha}}$ for each $i = 1, \ldots, m$. Obviously, such a storage requires a lot of memory. By contrast, updating the entries of the matrix $\mathcal{J}_{\mu^{\alpha}}$ one by one not only requires less memory, but also improves the convergence of the algorithm [3].

Infinite Cost

If a problem lasts infinitely long, the cost to finish the problem may take an infinite value

$$\lim_{N\to\infty} \sum_{k=0}^{N} \mathop{E}_{w_k} \left\{ g(x_k, \mu^{o}(.), w_k) \right\} = \infty. \tag{4.23}$$

Obviously, infinite costs cannot be represented in a computer. However, a remedy to this problem is to subtract a constant value h from the cost-to-go function after each iteration. The dynamic programming iteration can be reformulated as follows:

$$\mathcal{H}(x,w)+h=\min_{\mu(.)}\left[g(x,\mu(.),w)+\underset{\bar{w}\sim\mathcal{P}(w)}{E}\left\{\mathcal{H}\left(f(x,\mu(.),w),\bar{w}\right)\right\}\right], \quad (4.24)$$

where h is redefined after each iteration of the stochastic dynamic programming algorithm such that $\mathcal{H}(x,w)$ stays in a valid range. Since minimization is relative, it is clear by intuition that the constant h does not affect the optimal control law that is found by the optimization. For a rigorous proof see [3].

Redundant calculations

During the execution of the stochastic dynamic programming algorithm the discrete sets X, U and W are not changed. Thus, the permutations of x, u and w remain the same over all iterations. This fact allows the response of the system to be evaluated prior to executing the stochastic dynamic programming algorithm, instead of re-evaluating the model function in each iteration. The response is simply stored in two look-up tables

$$x_{k+1}=F(x,u,w) \quad : \quad \mathbb{R}^P\times\mathbb{R}^Q\times\mathbb{R}^R\rightarrow\mathbb{R}^P, \quad (4.25)$$
$$g_k=G(x,u,w) \quad : \quad \mathbb{R}^P\times\mathbb{R}^Q\times\mathbb{R}^R\rightarrow\mathbb{R}, \quad (4.26)$$

where F represents the state dynamics, i.e., x_{k+1} for all possible discrete permutations of states, control inputs and disturbances, and G represents the stage cost.

Furthermore, since the values stored in F and G do not change over the course of the execution of the algorithm, any interpolation that is dependent on these values can be simplified. For example the cost-to-go function has to be evaluated at the values stored in F, i.e., $\mathcal{J}\left(x_{k+1}=F(.),w\right)$. In fact, any n-dimensional interpolation can be split up into two steps: i) determining the grid points relevant for the interpolation and their corresponding weights and ii) evaluating the weighted sum of the values at the relevant grid points. The first step can be finished prior to running the stochastic dynamic programming algorithm and thus has to be evaluated only once.

Table 4.2: Discretization of states and inputs.

Variable	Symbol	Range		# Pts.
State of energy	E_b	25–100	%	50
Engine speed	ω_g	0–2300	rpm	8
Engine coolant temperature	ϑ_e	20–90	°C	4
Vehicle speed	v	0–80	km/h	15
Requested power	P_{req}	-280–280	kW	30
Control input	u	1–8	[–]	8

4.2.4 Discretization

The algorithm described above was applied to the vehicle model with the discretization of state, input and disturbance variables listed in Table 4.2. Obviously, there is a trade-off between number of discrete grid points in each dimension and the computational effort. Here, a reasonable trade-off was found by intuition.

4.2.5 Additional Penalty Factors

In the design of the controller, several additional penalty factors are used to trim the controller towards a behavior that is acceptable in a vehicle. These penalty factors are described below.

Buffer Limits

The controller automatically considers the upper limit of the buffer as a constraint that should not be violated, since such states are naturally penalized by the fact that braking energy is wasted and the overall fuel consumption is increased.

In order to enforce the lower limit of the buffer, an additional penalty factor is needed. The penalty is chosen linearly to the depth of the violation

$$g_1 = \kappa_{\text{low}} \cdot \max(0, E_{b,\min} - E_{b,k+1}) \tag{4.27}$$

The penalty is added to the cost functional as in (4.14). If the penalty is assigned a low value, the resulting controller maximizes the use of the energy buffer. However, the probability of the buffer being discharged completely during an acceleration is high. In this case the traction power is limited to the maximum power of the engine generator set, which is much lower than the nominal power of the traction machine, and thus this state is not desirable. In contrast, a high value leads to a controller that always starts to recharge the buffer way before the lower limit is reached, and thus it does not make the best use of the available buffering capacity.

In fact, this penalty factor trades off good fuel economy against the power reserve available to the driver. Since the parameter influences fuel economy significantly, its value needs to be chosen carefully for each combination of vehicle, driving mission and driving data used for controller synthesis.

Recharging during Standstill

In the real vehicle, stopping the engine generator unit takes several seconds. Therefore, the engine may be still running while the vehicle has already arrived at a standstill. In such a situation it is optimal in the stochastic sense to recharge the buffer to 100%. This fact is explained by considering that on the one hand, the vehicle continues with an acceleration after the stop, which implies a zero probability of wasting recuperation energy in the future. On the other hand, the probability of violating the lower limit of the buffer in the future is in fact quite high. Thus, the optimal action to minimize the probability of running into the lower state constraint by fully recharging the buffer.

Typically, the controller chooses an operating point close to the highest level of efficiency of the engine generator unit, which results in a rather high rotational speed of the engine. However, running the engine at high speed during standstill is not acceptable in a city bus. Therefore, an additional penalty factor that limits the engine generator unit to operating points with a low rotational speed is introduced

$$g_2 = \begin{cases} \kappa_{\text{chg}} & \text{if } (v_k < 20^{\text{km}}/_{\text{h}}) \cap (P_{\text{req},k} < 20\text{kW}) \cap (P_{g,k} > P_{g,k-1}) \\ 0 & \text{else.} \end{cases} \tag{4.28}$$

This factor penalizes any control input that leads to an increase of the generator output power (and thus of the speed of the engine generator unit) if the vehicle speed and power are low.

Since the conditions for this penalty are very specific, κ_{chg} can be chosen to take a very high value without the risk of deteriorating the performance of the controller. Thus the penalty factor does not require

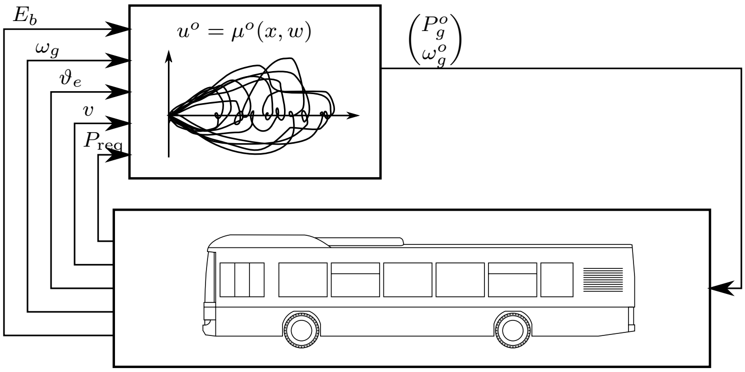


Figure 4.3: Online implementation of the stochastic dynamic programming controller.

a specific tuning process. Note that this factor penalizes engine starts at low vehicle speed and low traction power as well.

Engine Start/Stop

In order to avoid engine stops under cold conditions, the following penalty term is added to the cost functional:

$$g_3 = \begin{cases} \kappa_{\text{stop}} & \text{if } (\vartheta_{e,k} < 68°C) \cap (P_{g,k-1} > 0) \cap (P_{g,k} = 0) \\ 0 & \text{else.} \end{cases} \tag{4.29}$$

Again, the penalty factor is very specific and thus does not require a specific tuning process. The factor κ_{stop} can be assigned a very high value.

4.3 Online Algorithm

Figure 4.3 shows how the resulting control strategy μ is to be used on a vehicle. The straightforward way for such an online implementation, would be to simply use the control policy as a look-up table in function of the measured parameters

$$\begin{pmatrix} P_{g,k+1}^o \\ \omega_{g,k+1}^o \end{pmatrix} = \mu(x_k, u_k, w_k) = \mu^o(E_{b,k}, \omega_{g,k}, \vartheta_{e,k}, v_k, P_{\text{req},k}). \tag{4.30}$$

However, since the optimal control policy μ^o is generally a non-smooth function, interpolation does not generally lead to the optimal control input, especially when a rather sparse grid, such as the one listed in Table 4.2 is used.

The mathematically correct implementation would be to evaluate the vehicle model using all possible control inputs at each timestep, followed by a choice of the one control input that minimizes the right-hand side of (4.17). However, this would require both additional computational resources and storage capacity because both the model and the Markov chain would have to be stored in and evaluated by the onboard control unit. Obviously, those extras should be avoided. With the controller structure used in the vehicle, the control policy look-up table can be transformed in a way such as to allow for a smooth interpolation as follows:

Instead of directly calculating the optimal control input to be applied in the next timestep, as in (4.30), the map μ^o may be partly inverted. This inversion is achieved by finding a limit $E_b^{\mathrm{lim},u}$ that, when undershot by E_b activates the corresponding input u, where $u \in \{1, 2, \ldots, Q\}$. Consider that the current energy level of the buffer is below the limit $E_{b,k} < E_b^{\mathrm{lim},u}$. In this case the controller chooses the one control input u with the highest value satisfying the above condition, i.e., $P_{g,k+1} = P_g^u$. Obviously, for this inversion to be valid, the condition $E_b^{\mathrm{lim},u+1} \leq E_b^{\mathrm{lim},u}$ must hold for all possible control candidates u.

Figure 4.4 shows the limits as functions of the requested power for four different scenarios. The two upper plots show the limits for a low vehicle speed of 17 km/h, while the two lower ones show those limits for a higher vehicle speed of 51 km/h. The two plots on the left-hand side show the limits for the case when the engine is currently off, the two on the right-hand side for the case where the engine is currently running at an electric output power of 90 kW. The shaded regions indicate which control input is optimal. The darker the color, the higher the output power of the engine generator unit.

Consider the graph on the lower right-hand side. For a positive traction power, the controller expects an acceleration to a higher velocity. Therefore, the lower the energy level of the buffer, the higher a value of the engine power is requested, e.g., for $P_{\mathrm{req}} = 100\,\mathrm{kW}$, the engine is turned off for an energy level of the buffer above 530 Wh, while for lower energy levels, the engine power is increased step by step.

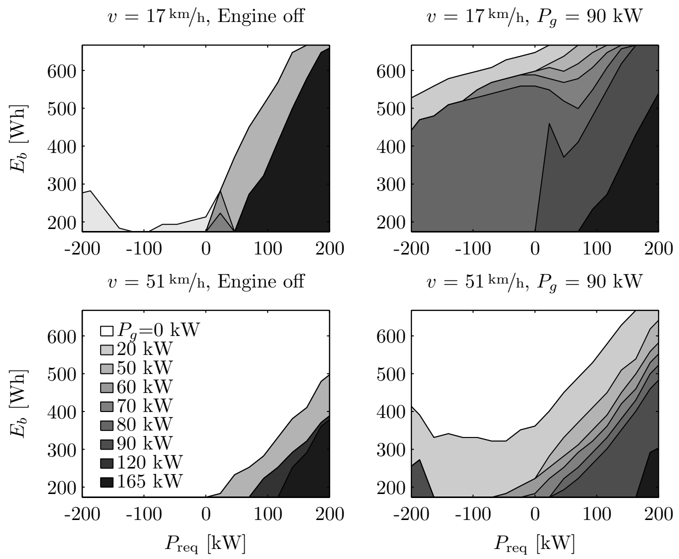


Figure 4.4: Visualization of the control input map obtained from stochastic dynamic programming.

In certain situations the controller might exhibit a counterintuitive behavior, e.g., at 200 kW of braking power at a rather low vehicle speed (see upper left graph). The reason for this behaviour is explained by the fact that these situations cannot occur in practise, as well as in the data set used to generate the Markov chain. Therefore, the Markov chain was not properly identified for these subsets of W. However, since these scenarios cannot occur in practice, they are irrelevant.

4.3.1 Controller Synthesis

When the proposed stochastic dynamic programming algorithm is used to synthesize an energy management controller, the resulting control law is specific to a certain vehicle, driver and driving mission. A general baseline controller can be found by using several different data sets, containing a number of driving missions to generate the Markov chain. Additionally, it is possible to generate controllers that are specific to a certain mission or bus driver. In the results section it is demonstrated that such specific controller exhibits a better performance than the baseline controller.

4.4 Results

A baseline controller has been synthesized using data from several different driving missions. The goal was to find a good, general purpose baseline controller that can be used if the route is not known in advance. This baseline controller was implemented on the electronic control unit of the bus shown in Figure 1.1. After a relatively short test phase that was required to tune the penalty factors and ensure good driveability, the fuel consumption of the bus was evaluated in a practical test via the Standardized On Road Test cycles [52]. The test is comprised of three regulatory test cycles that correspond to SORT 1—*heavy duty inner city*, SORT 2—*light city* and SORT 3—*urban operation*. In the test, only the fuel consumption values corresponding to SORT 1 and 2 were evaluated.

For the SORT 2 test cycle, Figure 4.5 shows the velocity trajectory driven, together with the requested power, the generator power and the energy level of the supercapacitor. The driving of the bus was automated in order to exclude the influence of the driver. To follow the reference speed profile, a PI controller with feedforward action was used. Over the last few meters before every stop, the driver was commanded to use the service brakes to exactly position the vehicle at each stop.

The fuel consumption was measured using a fuel flow meter as well as an estimation based on injection timing from the engine control unit. For calculating reliably fuel economy values, each of the SORT test cycles was driven at least ten times. Differences between the initial and the final energy level of the buffer were compensated for by evaluating equivalent fuel consumption values.

4.4.1 Comparison to a Diesel Bus

Figure 4.6 shows a comparison of the fuel consumption results obtained from measurements conducted using both the HESS Hybrid and a conventional benchmark vehicle with similar specifications. In a heavy traffic inner city scenario, fuel consumption is reduced by more than 27% compared to the conventional diesel bus. In a light city scenario, the savings are about 22.6%. The urban operation contains less stop-and-go traffic and longer phases of constant speed, causing the fuel saving potential to be somewhat lower, but still at a substantial 17.5%.

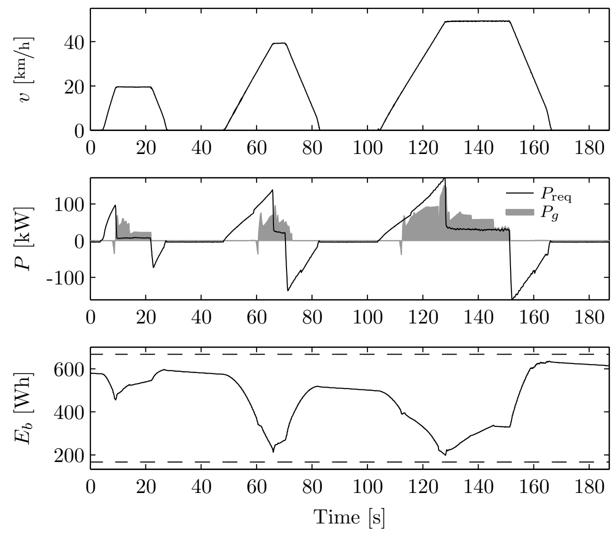


Figure 4.5: A single repetition of the SORT 2 test cycle. The graph shows the velocity profile (upper graph) together with the requested and the generator power (middle graph) and the energy level of the buffer (lower graph).

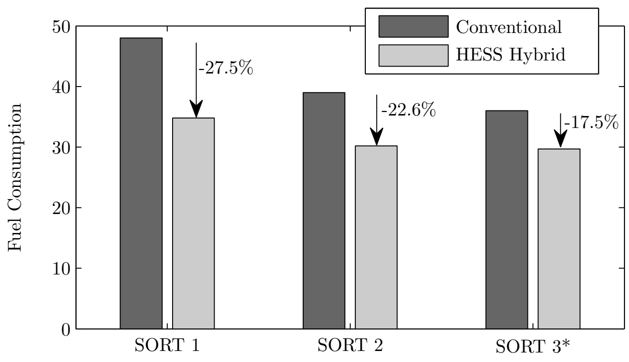


Figure 4.6: Comparison of the fuel consumption results of a conventional diesel bus and the serial hybrid bus. The result on SORT 3 is simulated on a validated vehicle model.

Table 4.3: Comparison of fuel consumption results on the SORT cycles using the baseline controller and the global optimal energy management. Values are given in l/100km.

	measured	simulated	optimal	diff.
SORT 1	34.8±1.2	34.97	33.93	+3.07%
SORT 2	30.2±0.8	30.08	29.23	+2.91%
SORT 3	–	29.70	28.36	+4.72%

4.4.2 Optimality

In order to judge the quality of the controller, the results are compared to those of the best possible controller, namely, the globally optimal solution. For this comparison a validated model is needed. This comparison is shown in Table 4.3 which contains three columns. The first column summarizes the measured fuel consumption in l/100km for SORT 1 and 2, together with the standard deviation of the measurement. The second column shows the simulated fuel consumption on the SORT cycles using a validated model and the strategy used in the experiments. Finally, the third column shows the globally optimal fuel consumption simulated on the same validated model, however using the optimal strategy calculated by means of the deterministic dynamic programming algorithm.

Several observations can be made. First, the error between practical results and simulation (columns 1 and 2) is well inside the tolerance of the measurement, which speaks for a valid model. Second, the difference between the baseline controller and the global optimal solution is smaller than 3% for SORT 1 and 2, and below 5% for SORT 3. In other words, any measure to improve the controller can reduce the fuel consumption by 3% to 5% maximally.

Compared to the optimal solution, the online controller leads to a slightly increased number of engine starts (2.9 starts per repetition of the SORT 1 test cycle instead of 2.1) and therefore to a slightly reduced overall efficiency of the engine generator unit (37.8% instead of 38.5%), which is the reason for the increased fuel consumption.

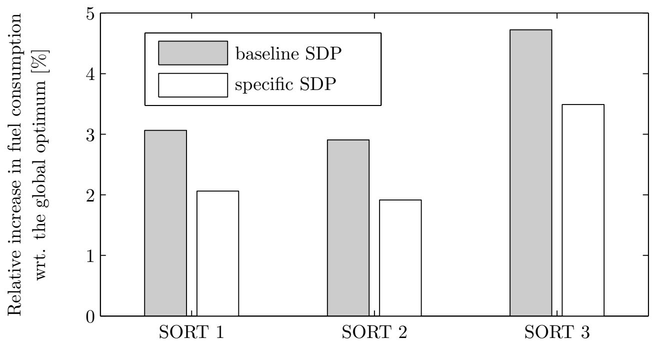


Figure 4.7: Relative excess consumption with respect to the global optimal solution obtained by dynamic programming.

4.4.3 Mission Specific Controller

The measurements on the SORT cycles were conducted using a general purpose baseline controller. However, with the stochastic dynamic programming algorithm it is possible to design specific controllers for each of the SORT cycles, which may lead to a further reduction of fuel consumption. Figure 4.7 shows the results of a small simulation study. The vehicle driving the three SORT cycles was simulated three times: once using the baseline controller that was used for the on-road test, once using the cycle-specific controllers, and finally, once using the global optimal strategy obtained by dynamic programming.

Figure 4.7 shows that the baseline controller achieves close-to-optimal results already. On the SORT 1 and 2 test cycles, the fuel consumption is increased by about 3% compared to the theoretical global optimum. If a cycle-specific controller is used, this difference can be further reduced to about 2%. On the SORT 3 test cycle, a similar benefit of a specific controller can be expected.

Compared to the baseline controller, the use of the specific controller leads to a slightly better overall efficiency of the engine generator unit (38.1% instead of 37.8%) and of the supercapacitor (97.4% instead of 97.1%). Table 4.4 shows the same results as Figure 4.7 in tabulated form.

Table 4.4: Comparison of the fuel consumption results on the SORT cycles using a specific controller. Values are given in $^{l}/_{100\text{km}}$.

	specific controller	optimal	diff.
SORT 1	34.63	33.93	+2.06%
SORT 2	29.79	29.23	+1.92%
SORT 3	29.35	28.36	+3.49%

4.5 Conclusion

This chapter introduced an efficient algorithm that rapidly solves the stochastic optimal energy management problem of a serial hybrid electric bus. The result is a stationary control law that can be easily implemented on an onboard electronic control unit. The design scheme allows the optimal tradeoff to be found among several objectives, such as good fuel economy and the amount of power reserve. Furthermore, the controller permits additional system variables to be monitored, such as the engine coolant temperature. The behavior of the controller has been well received by both passengers and drivers. The resulting fuel consumption has been shown to be close to the one obtained by dynamic programming (3-5 % increase with respect to dynamic programming) on flat, repetitive driving cycles. Furthermore, a small improvement can be achieved by designing a specific controller for each bus line. The performance of the proposed controller on a more realistic driving cycle is described in the next chapter.

Chapter 5

Predictive Energy Management

The previous chapter introduced an online energy management controller based on stochastic dynamic programming. The controller has been shown to deliver close to optimal results in practice when tested using the Standardized On-Road Test cycles. However, these test cycles arguably do not represent realistic operating conditions of a bus.

Therefore, in this chapter, the performance of the controller is tested on a more aggressive bus line with a pronounced altitude profile and with prolonged phases of driving the bus at its power limits (Line 4 from Chapter 3). In order to ensure a good comparability of the results, the tests are conducted on a test bench. The performance of the baseline controller is no longer close to optimal on this line, which can be explained by the fact that the controller causes a dissipation of braking energy in the braking resistor during the braking phase before stops or when driving downhill.

This is a well known problem with energy management strategies that rely on real-time available information only. To circumvent this problem, several *predictive* energy management controllers have been proposed in the literature. Such predictive controllers usually combine satellite positioning with some kind of map to provide the energy management controller with information about the route ahead of the vehicle. Some authors use deterministic dynamic programming in a model-predictive control fashion [53, 26], some use dynamic programming to estimate the value of the equivalence factor for the use in the equivalent consumption minimization strategy [54]. Others directly adapt the equivalence factor using heuristics [55, 56], pattern recognition algorithms [57], stochastic driver models [58] or PI-feedback control [59] together with a state-of-charge reference profile that has been optimized using quadratic programming [60] or dynamic programming [61].

Analogous to findings presented in the literature, the energy losses observed in our experiments with the serial hybrid electric bus are identified to be caused by the fact that the baseline controller is not aware of future driving conditions, such as future braking events or up/downhill sections of the bus line. In the case of a city bus, the driving mission is perfectly known in advance such that, together with satellite navigation, providing trip-specific information to the energy management controller is possible.

This chapter introduces a predictive extension to the baseline controller. The predictive extension recovers a large portion of the optimality while using only a small amount of predictive information. Compared to the global optimal solution, the predictive extension reduced the excess consumption from 6.7% to 1.2% in the experiment.

5.1 Testbench

The predictive controller has been tested on a testbench, which greatly simplifies its implementation. In an on-vehicle test, a satellite positioning receiver and an algorithm that extracts the predictive data from a map would be needed. On the testbench these subsystems were replaced by look-up tables.

The testbench is shown in Figure 5.1. It includes all powertrain components, together with their corresponding inverters and low-level controllers, i.e., the engine generator unit, the electric traction motor, the supercapacitor, the braking resistor and the DC link. The vehicle chassis has been replaced by an electric testbench machine that emulates the forces acting on the crankshaft of the traction machine. The vehicle/driver emulation is detailed in the following subsection. The low level controllers for the DC link, the supercapacitor, the engine generator unit etc., as well as the energy management controller are implemented on their corresponding onboard control units that have been installed on the testbench as well.

5.1.1 Vehicle and Driver Emulation

Figure 5.1 shows the vehicle and driver emulation that was implemented on the testbench computer. Given a certain time t and a certain driven distance s, the first block determines the vehicle speed setpoint and the current road inclination. The second block emulates the driver, who is approximated by a PI controller tracking the desired velocity profile. The

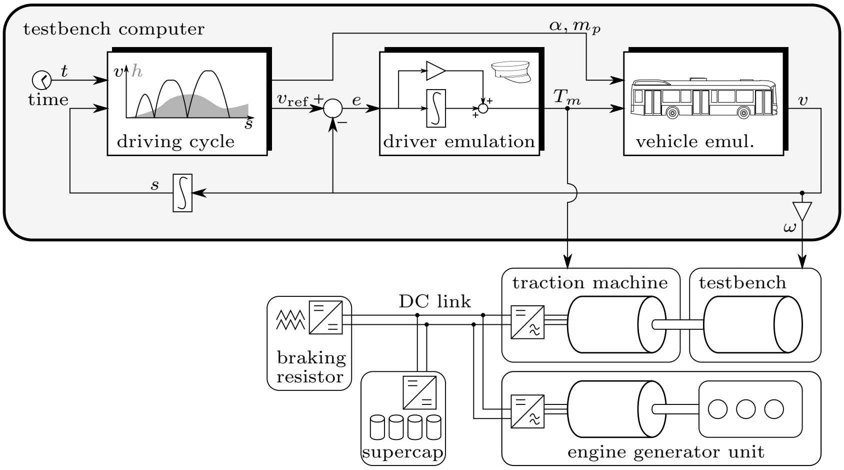


Figure 5.1: Scheme of the testbench and the vehicle/driver emulation.

driver output is a torque setpoint that is communicated to and realized by the low-level controller of the real traction motor mounted on the testbench. The last block emulates the forces currently acting on the vehicle body and calculates the resulting acceleration and vehicle speed. This information is then communicated to and realized by the testbench machine and its corresponding low-level controller. Furthermore, the vehicle speed calculated by the emulation block is fed back to the driver emulation. The vehicle speed signal serves to calculate by integration the distance driven.

5.1.2 First Results

The testbench is used to emulate the vehicle driving Line 4, shown in Figure 5.2. The fuel consumption of the engine is measured using a fuel flow meter. The result is shown in Table 5.1, together with two computer simulations. The first simulation includes the baseline controller, while the second simulation includes the global optimal energy management strategy that was evaluated using dynamic programming. Table 5.1 the error between measurement and simulation is small (0.6%), which speaks for a valid model allowing the comparison to the optimal solution. Furthermore, the performance of the baseline controller is suboptimal (+6.7% increase).

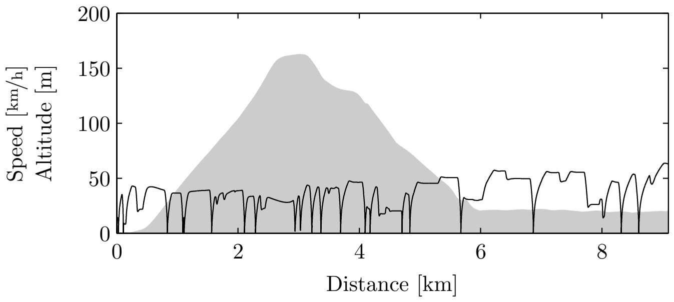


Figure 5.2: Speed and altitude profile vs. distance of Line 4.

Table 5.1: Measured and simulated fuel consumption results on Line 4.

measurement	50.93	l/100km
simulation	50.62	l/100km
optimal solution	47.44	l/100km
increase vs. optimal	+6.7	%

The increase in fuel consumption with respect to the optimal solution is explained by the fact that the baseline energy management controller does not include any information specific to the velocity driven and the altitude profile. Compared to the globally optimal solution, this controller results in an increased usage of the braking resistor (3.0 kWh instead of 2.6 kWh) and a slightly decreased efficiency of the engine generator unit (32.8% instead of 34.5%). Overall, these effects result in a suboptimal performance of the baseline controller.

5.2 Predictive Energy Management Controller

The idea pursued in this chapter is to find a predictive extension to the baseline controller that recovers optimality by taking into account some predictive information about the bus line. Using satellite navigation, the current position of the bus can be detected, see Figure 5.3. Together with a map stored on the onboard control unit, it is possible to provide the controller with position dependent information about the bus line, such

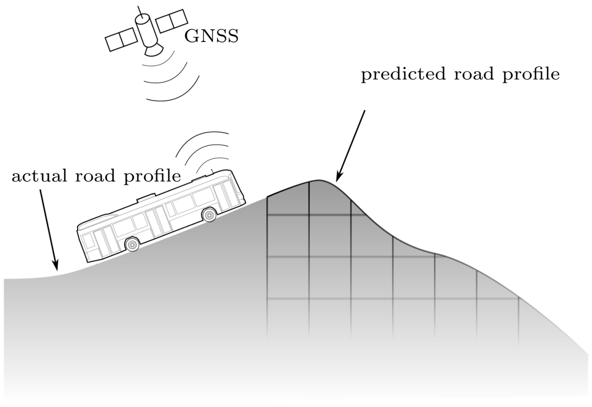


Figure 5.3: Road profile prediction.

as the future elevation profile or the distance to the next stop. The main benefit of such a system is that the optimality of the energy management controller performance can be recovered to a large extent. Furthermore, the system can automatically shut off the engine generator unit at bus stops in order to reduce noise levels. Another advantage is that the system can recharge the buffer prior to a steep uphill section, where the bus is likely to be operated at its power limits.

The predictive extension is shown in Figure 5.4. Basically, the extension calculates the energy to be delivered/absorbed by the energy buffer until the next critical point where the buffer limits are likely to be violated. Since the energy capacity of the supercapacitor is rather limited the next critical point is usually the next stop, where all kinetic energy of the vehicle should be used to recharge the buffer. Based on this energy request, an upper and a lower bound for the current buffer energy level is calculated. If the control actions of the baseline controller lead to a violation of the upper or the lower bound in the future, the generator power is limited by the predictive extension. If there is no violation, the predictive extension remains inactive.

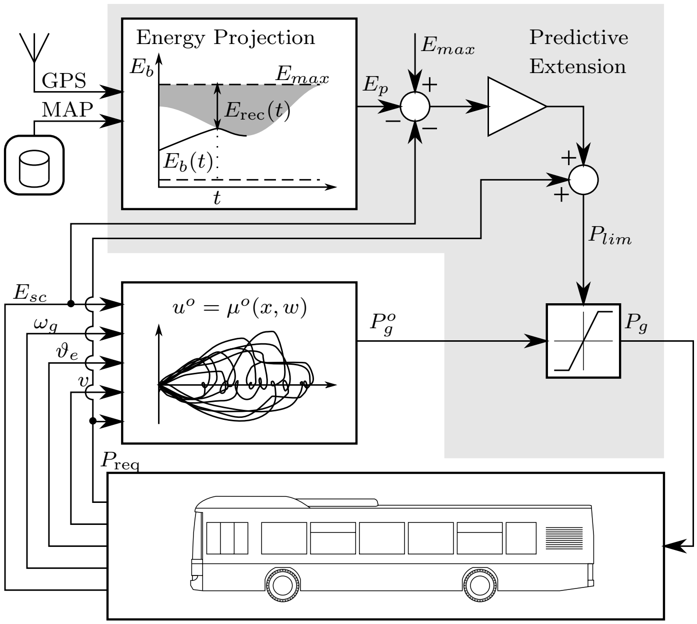


Figure 5.4: Predictive extension to the SDP controller.

5.2.1 Relevant Information

The energy projection requires some predictive information about the busline. In order to keep the system as simple as possible, the algorithm is designed to use i) as little information as possible and ii) only non-probabilistic information. .

Since the vehicle is equipped with supercapacitors, the energy buffering capacity is very limited. The baseline controller often cannot prevent braking energy from being wasted during the last braking phase before a stop. The main purpose of the predictive extension is thus to avoid such wasting events. This chapter proposes that this goal can be achieved by using just the information concerning

- the distance to the upcoming stop s_k,
- the absolute maximum altitude variation within 200 m Δh_k,
- and the actual velocity v_k.

5.2.2 Projection of Energy

Based on this information, the energy to be delivered/absorbed by the supercapacitor in the future can be estimated using a static relationship. For instance, the energy to be absorbed by the supercapacitor is

$$\begin{aligned} E_{\mathrm{rec},k} &= E_{\mathrm{kin}} + E_{\mathrm{pot}} - E_{\mathrm{r}} - E_{\mathrm{a}} \\ &= \eta \frac{1}{2} m v_k^2 + \eta^{-sgn(\Delta h_k)} mg\Delta h_k - c_r mg s_k - \frac{1}{2}\rho_a A_f c_d v_k^2 s_k, \end{aligned} \tag{5.1}$$

where E_{kin} and E_{pot} are the recoverable parts of the kinetic and potential energy currently stored in the vehicle body and E_{r} and E_{a} are the amounts of energy required to overcome rolling and aerodynamic friction. The quantity $E_{\mathrm{rec},k}$ is positive if the supercapacitor needs to absorb energy. Since it represents the minimum amount of energy that needs to be absorbed, the generator power is assumed to be zero until the next stop. The following assumptions were made in this relationship:

1. The efficiency of the traction machine and its power electronics is constant. Due to the conversion efficiency being high throughout the operating range of the traction machine, this assumption introduces only a small error.

2. The aerodynamic friction force remains constant from the actual position of the vehicle to the next stop. This assumption allows the prediction to be conducted without the precise knowledge about the future velocity profile. With a heavy vehicle driving at low velocity, the aerodynamic friction is much smaller than the rolling friction, and thus the error introduced by this assumption is almost negligible.

Note that the absolute maximum altitude variation is taken over a fixed horizon rather than only up to the next stop. By this, up- or downhill sections that lie beyond the next stop can be accounted for.

When the energy to be delivered by the supercapacitor until the next stop is estimated the generator power needs to be considered as well:

$$\begin{aligned} E_{\mathrm{trc},k} &= E_{\mathrm{r}} + E_{\mathrm{a}} - E_{\mathrm{pot}} - E_{\mathrm{kin}} - E_{\mathrm{g}} \\ &= c_r mg s_k + \frac{1}{2}\rho_a A_f c_d v_k^2 s_k - \eta^{-sgn(\Delta h_k)} mg\Delta h_k - \eta \frac{1}{2} m v_k^2 \\ &\quad - P_g^{\mathrm{best}} \left(\frac{s_k}{v_k} - t_{\mathrm{start}} \Big(1 - e(t)\Big) \right). \end{aligned} \tag{5.2}$$

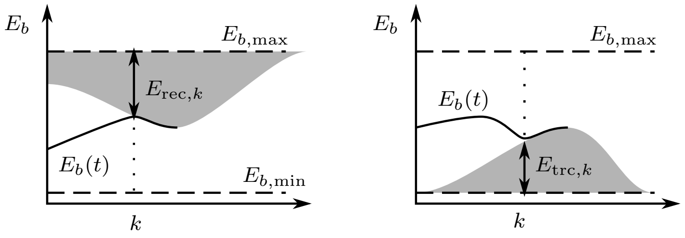


Figure 5.5: Energy projections for the cases $E_{\mathrm{rec},k} > 0$ (left) and $E_{\mathrm{trc},k} > 0$ (right), respectively.

Here, E_g is the maximum amount of energy that can be delivered by the engine generator unit, considering that it is running all the time until the next stop at its best efficiency operating point P_g^{best}. The term $\frac{s_k}{v_k}$ estimates the time left before the next stop is reached. The variable $e(t) \in \{0, 1\}$ is a boolean variable representing the current on/off state of the engine. Its value is one if the engine is running, and zero if the engine is shut off. The expression in brackets thus represents the time left to produce energy until the next stop is reached. By multiplying this value with the fuel optimal electric power of the engine generator unit, an estimate is found of the maximum amount of energy that can be generated until the next stop is reached. The quantity $E_{\mathrm{trc},k}$ is positive if the supercapacitor needs to deliver energy.

The remaining parts of the approximation remain the same as in (5.1).

5.2.3 Buffer Energy Limits

With the estimated amounts of energy E_{rec} and E_{trc} it is possible to calculate a maximum and a minimum energy level that should not be exceeded. Figure 5.5 shows examples for the two cases where either $E_{\mathrm{rec},k} > 0$ (left) or $E_{\mathrm{trc},k} > 0$ (right). The current minimum buffer energy level $E_{\mathrm{p},k}^{\min}$ (subscript p stands for *predicted*) is obtained by adding $E_{\mathrm{trc},k}$ to the minimum allowed energy level $E_{b,\min}$

$$E_{\mathrm{p},k}^{\min} = E_{b,\min} + \max\{E_{\mathrm{trc},k}, 0\} \tag{5.3}$$

$$E_{\mathrm{p},k}^{\max} = \max\left\{E_{b,\max} - \max\{E_{\mathrm{rec},k}, 0\}, E_{\mathrm{p},k}^{\min}\right\}. \tag{5.4}$$

The second equation is analog to the first one but furthermore, it ensures that $E_{\mathrm{p},k}^{\max}$ cannot take any values smaller than $E_{\mathrm{p},k}^{\min}$.

5.2.4 Predictive Control Action

The predictive extension adjusts the control action of the basic controller, if it leads to a violation of one of the limits $E_{\mathrm{p},k}^{\min}$ and $E_{\mathrm{p},k}^{\max}$ only. If the upper limit is violated the generator power needs to be reduced, while if the lower limit is violated the generator power needs to be increased. A proportional controller with a gain κ and with P_{req} as a reference serves to calculate an upper and a lower bound on the generator power

$$P_{\mathrm{lim},k}^{\max} = P_{\mathrm{req},k} - \kappa(E_{b,k} - E_{\mathrm{p},k}^{\max}) \tag{5.5}$$

$$P_{\mathrm{lim},k}^{\min} = P_{\mathrm{req},k} + \kappa(E_{\mathrm{p},k}^{\min} - E_{b,k}) \tag{5.6}$$

These values are then used to limit the control action of the basic controller, if necessary

$$P_{g,k} = \max\Big\{ \min\{P_g^o, P_{\mathrm{lim},k}^{\max}\}, P_{\mathrm{lim},k}^{\min} \Big\}. \tag{5.7}$$

The gain κ determines by how much the control input is adjusted.

5.3 Results

As depicted in Figure 5.2, Line 4 once again was emulated on the testbench, using the baseline controller together with the predictive extension. For the downhill section, Figure 5.6 shows a comparison of the emulations with and without the predictive extension, i.e., *with preview* in green and and *without preview* in red. The upper graph shows the speed and altitude profiles, while the lower graph shows the buffer energy trajectories. Clearly the predictive controller minimizes the dissipation of energy in the braking resistor (between 3200 and 3400 m) by shutting off the engine earlier (around 2950 m). This action causes the overall energy consumption to be minimized. Both controllers follow the reference speed profile (black line) equally well.

Figure 5.7 shows an uphill section in the same way as Figure 5.6. Here, the predictive extension makes use of the stops (around 800 and 1040 m) to recharge the buffer. These recharges cause the supercapacitor to not be fully discharged during driving (e.g., between 1100 and 1400 m) and thus to still have enough power to precisely follow the desired velocity

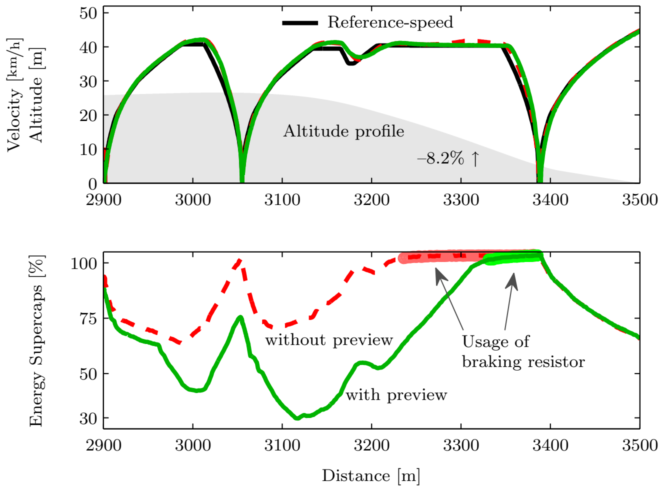


Figure 5.6: Comparison of the predictive (green) and the non-predictive (red) energy management controller on a downhill section.

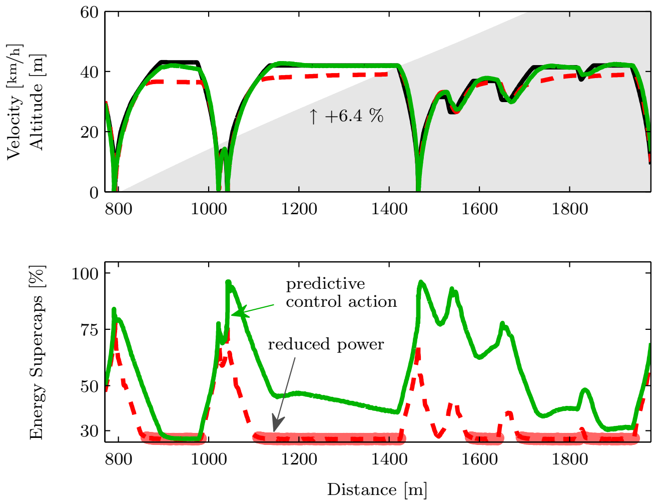


Figure 5.7: Comparison of the predictive (green) and the non-predictive (red) energy management controller on an uphill section.

Table 5.2: Measured and simulated fuel consumption results on Line 4.

	without preview	with preview	
measurement	50.93	50.54	l/100km
simulation	50.62	50.55	l/100km
optimal	47.44	49.97	l/100km
increase vs. optimal	+6.7	+1.2	%

profile. Without the predictive extension, the buffer would tend to be fully discharged because the baseline controller is not aware of the steep uphill part following each of the stops. Therefore, without the predictive extension, it is not possible to follow the desired velocity profile exactly.

5.3.1 Comparison to the Optimal Solution

Since the emulations are of the dynamic type rather than of the quasi-static type, the driven velocity profile is influenced by control constraints, such as the maximum power of the supercapacitors. As Figure 5.7 shows, in the case without preview, the bus is not able to follow the desired velocity profile exactly. However, instead of accumulating the delays caused by driving more slowly, the delays are compensated for by shortening the halting time at the stops. Overall, this compensation results in a lower mechanical energy demand due to lower friction, which in turn prohibits a direct comparison of fuel consumption results.

Therefore, the performance of the controller is once again evaluated by a simulation on the validated vehicle model. The global optimal solution is obtained with dynamic programming. Table 5.2 summarizes the results. While the baseline controller alone results in an increase of 6.7% in the fuel consumption with respect to the optimal solution, the predictive extension is able to recover the optimality of the control performance to a large extent. With the predictive extension, the fuel consumption is only 1.2% higher than with the global optimal solution.

5.4 Conclusion

This chapter analyzed the performance of the baseline energy management controller based on stochastic dynamic programming that has been shown to deliver close-to-optimal results on the Standardized On-Road Test cycles. On a more realistic driving profile, including an aggressive altitude profile and prolonged phases of driving at the power limits of the bus, the control performance of the baseline controller was no longer close to optimal. The sub-optimality has been identified to be caused by a loss of braking energy during the last braking phase before a stop and during the downhill sections of the driving cycle. Therefore, a predictive extension to the baseline controller has been proposed that uses a minimal amount of predictive information. The predictive extension restores close-to-optimality, by preventing energy wasting events.

Future work includes testing the baseline controller and the predictive extension in practice. An extensive test phase with a duration of six months is scheduled to begin shortly after this thesis is published. Within this test phase, the bus will be run by a public transportation operator in a German city.

Summary

The component sizing problem studied in the first part of this thesis is usually adressed in a heuristic approach, in which the drivetrain components such as engine, electric traction motor, and energy buffer are dimensioned according to certain heuristic sizing rules. However, as it is described in Part I, the problem of component sizing is closely coupled to the energy management problem. Thus, an integrated design approach is necessary to obtain globally optimal results. Two numerical methods have been investigated that solve the optimal control problem corresponding to the energy management of a serial hybrid electric bus. The first one, dynamic programming, is generic and can theoretically be applied to any kind of hybrid powertrain. The vehicle model can be of any type, even non-linear, or including discrete state and/or decision variables. The drawbacks of this method are its high computational burden and the fact that the computational effort increases exponentially with the number of dimensions. For instance, optimizing the energy management for a bus with both batteries and supercapacitors requires a multiple of the evaluation time needed for a single-buffer powertrain. Therefore, another methodology based on convex optimization was developed. The required model approximations have been shown to introduce moderate errors. In the case of a serial hybrid powertrain, a method has been proposed that includes the discrete engine on/off decision into the convex problem. Future work in this direction is to investigate how this methodology can be extended to other topologies, such as the parallel hybrid powertrain.

The second part of this thesis was devoted to implementing an online energy management controller. First, a generic method was investigated to find an energy management controller that performs optimally in the probabilistic sense. The method delivers close-to-optimal results on the Standardized On-Road Test cycles. Furthermore, the controller is easy to trim towards a behavior that is acceptable in a city bus, e.g., avoiding high noise levels at a stop or avoiding engine shut-offs when the engine coolant

temperature is low. After this initial test phase, the performance of the controller was tested on a more aggressive driving cycle using a testbench emulation. Since the controller does not include any information specific to the bus line, it may happen that the buffer is being fully charged during the last braking phase before a stop or during a downhill section. Thus, the excess energy from recuperation has to be dissipated in the braking resistor. These so-called energy wasting events can be avoided using a predictive extension that takes into account specific route information. The extension uses a minimum amount of predictive information that can be easily stored in a map of the bus line. For an online implementation, a global positioning system sensor is needed to determine the actual position of the bus. On the testbench, close-to-optimal results were achieved when using the predictive extension together with the baseline controller. Furthermore, drivability is improved because the buffer is being recharged prior to steep uphill sections of the busline, where the bus would run into its power limits otherwise.

Future work includes the implementation of the predictive controller on the bus. During a six-month test phase, the bus is to be run by a public transportation operator in a German city.

* * *

Bibliography

[1] L. Guzzella and A. Sciarretta, *Vehicle Propulsion Systems: Introduction to Modeling and Optimization*, 2nd ed. Berlin: Springer Verlag, 2007.

[2] R. Bellman, *Dynamic Programming.* Princeton, NJ: Princeton University Press, 1957.

[3] D. Bertsekas, *Dynamic Programming and Optimal Control.* Belmont, MA: Athena Scientific, 1995.

[4] O. Sundström, D. Ambühl, and L. Guzzella, "On implementation of dynamic programming for optimal control problems with final state constraints," in *Proceedings of Les Rencontres Scientifiques de l'IFP: Advances in Hybrid Powertrains*, Rueil-Malmaison, France, November 2008.

[5] P. Varaiya, "Reach set computation using optimal control," in *Proceedings of the KIT Workshop on Verification of Hybrid Systems*, Grenoble, France, 1998, pp. 377–383.

[6] A. Kurzhanski and P. Varaiya, "Dynamic optimization for reachability problems," *Journal of Optimization Theory and Applications*, vol. 108, pp. 227–251, 2001.

[7] I. Mitchell, A. Bayen, and C. Tomlin, "Validating a hamilton-jacobi approximation to hybrid system reachable sets," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science. Springer Verlag, 2001, vol. 2034, pp. 418–432.

[8] R. Luus, *Iterative Dynamic Programming*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 2000.

[9] F.-Y. Wang, H. Zhang, and D. Liu, "Adaptive dynamic programming: An introduction," *Computational Intelligence Magazine, IEEE*, vol. 4, no. 2, pp. 39 –47, may 2009.

[10] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality.* Newark: Wiley, 2007.

[11] D. Bertsekas and J. Tsitsiklis, "Neuro-dynamic programming: an overview," in *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, vol. 1, dec 1995, pp. 560 –564 vol.1.

[12] R. E. Larson, *State Increment Dynamic Programming.* American Elsevier Pub. Co., 1968.

[13] M. Back, S. Terwen, and V. Krebs, "Predictive powertrain control for hybrid electric vehicles," in *IFAC Symposium on Advances in Automotive Control*, Salerno, Italy, April 2004, pp. 451–457.

[14] O. Sundström and L. Guzzella, "A generic dynamic programming matlab function," in *Control Applications, (CCA) Intelligent Control, (ISIC), 2009 IEEE*, July 2009, pp. 1625 –1630.

[15] O. Sundström, L. Guzzella, and P. Soltic, "Optimal hybridization in two parallel hybrid electric vehicles using dynamic programming," in *Proceedings of the 17th IFAC World Congress*, Seoul, Korea, July 6–11 2008, pp. 4642–4647.

[16] C. Lin, Z. Filipi, Y. Wang, L. Louca, H. Peng, D. Assanis, and J. Stein, "Integrated, feed-forward hybrid electric vehicle simulation in simulink and its use for power management studies," 2001.

[17] M. Back, M. Simons, F. Kirschbaum, and V. Krebs, "Predictive control of drivetrains," in *Proc. of the 15th IFAC World Congress in Barcelona*, 2002.

[18] L. V. Pérez, G. R. Bossio, D. Moitre, and G. O. Garcia, "Optimization of power management in an hybrid electric vehicle using dynamic programming," *Mathematics and Computers in Simulation*, vol. 73, no. 1-4, pp. 244 – 254, 2006, applied and Computational Mathematics - Selected Papers of the Fifth PanAmerican Workshop - June 21-25, 2004, Tegucigalpa, Honduras.

[19] O. Sundström, L. Guzzella, and P. Soltic, "Torque-assist hybrid electric powertrain sizing: From optimal control towards a sizing law," *IEEE Transactions on Control Systems Technology*, vol. Accepted for publication, 2009.

[20] H. P. Geering, *Optimal Control with Engineering Applications.* Berlin, Germany: Springer Verlag, 2007.

[21] A. Sciarretta and L. Guzzella, "Control of hybrid electric vehicles," *IEEE Control Systems Magazine*, vol. 27, no. 2, pp. 60–70, 2007.

[22] N. Kim, S. Cha, and H. Peng, "Optimal Control of Hybrid Electric Vehicles Based on Pontryagin's Minimum Principle," *IEEE Transactions on Control Systems Technology*, vol. PP, no. 99, pp. 1–9, 2011.

[23] S. P. Boyd and L. Vendenberghe, *Convex Optimization.* New-York: Cambridge University Press, 2004.

[24] E. D. Tate and S. P. Boyd, "Finding Ultimate Limits of Performance for Hybrid Electric Vehicles," *SAE Transactions*, no. 00FTT-50, 1998.

[25] S. Terwen, M. Back, and V. Krebs, "Predictive powertrain control for heavy duty trucks," in *4th IFAC Symposium on Advances in Automotive Control, Salerno, Italy*, 2004.

[26] M. Koot, J. Kessels, B. de Jager, W. Heemels, P. van den Bosch, and M. Steinbuch, "Energy management strategies for vehicular electric power systems," *Vehicular Technology, IEEE Transactions on*, vol. 54, no. 3, pp. 771–782, May 2005.

[27] R. Beck, A. Bollig, and D. Abel, "Comparison of Two Real-Time Predictive Strategies for the Optimal Energy Management of a Hybrid Electric Vehicle," *Oil & Gas Science and Technology - Revue de l'IFP*, vol. 62, no. 4, pp. 635–643, 2007.

[28] N. Murgovski, L. Johannesson, J. Hellgren, B. Egardt, and J. Sjöberg, "Convex Optimization of Charging Infrastructure Design and Component Sizing of a Plug-in Series HEV Powertrain," in *Proc. IFAC World Congress*, 2011, pp. 13 052–13 057.

[29] N. Murgovski, L. Johannesson, J. Sjöberg, and B. Egardt, "Component sizing of a plug-in hybrid electric powertrain via convex optimization," *Mechatronics*, vol. 22, no. 1, pp. 106–120, Jan. 2012.

[30] N. Murgovski, L. Johannesson, and J. Sjoberg, "Engine on/off control for dimensioning hybrid electric powertrains via convex optimization," *Vehicular Technology, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2013.

[31] X. Wei, L. Guzzella, V. Utkin, and G. Rizzoni, "Model-based fuel optimal control of hybrid electric vehicle using variable structure control systems," *Journal of Dynamic Systems, Measurement and Control*, vol. 129, no. 1, pp. 13–19, 2007.

[32] D. Ambühl, O. Sundström, A. Sciarretta, and L. Guzzella, "Explicit optimal control policy and its practical application for hybrid electric powertrains," *Control Engineering Practice*, vol. 18, no. 12, pp. 1429–1439, 2010.

[33] N. Murgovski, L. Johannesson, and J. Sjöberg, "Convex modeling of energy buffers in power control applications," in *IFAC Workshop on Engine and Powertrain Control, Simulation and Modeling (E-COSM)*, 2012, pp. 92–99.

[34] Matlab, "7.14.0.739 (r2012a) 32 bit," Natick, Massachusetts, 2012.

[35] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming (web page and software)," *URL http://stanford.edu/boyd/cvx*, 2008.

[36] J. F. Sturm, "SeDuMi: version 1.05," 2004.

[37] H.-D. Lee, E.-S. Koo, S.-K. Sul, J.-S. Kim, M. Kamiya, H. Ikeda, S. Shinohara, and H. Yoshida, "Torque control strategy for a parallel-hybrid vehicle using fuzzy logic," *Industry Applications Magazine, IEEE*, vol. 6, no. 6, pp. 33–38, 2000.

[38] N. J. Schouten, A. Salman, and N. A. Kheir, "Energy management strategies for parallel hybrid vehicles using fuzzy logic," *Control Engineering Practice*, vol. 11, no. 2, pp. 171–177, 2003.

[39] K. Wipke, M. Cuddy, and S. Burch, “ADVISOR 2.1: a user-friendly advanced powertrain simulation using a combined backward/forward approach,” *IEEE Transactions on Vehicular Technology*, vol. 48, no. 6, pp. 1751–1761, 1999.

[40] C. Liang, W. Qingnian, L. Youde, M. Zhimin, Z. Ziliang, and L. Di, “Study of the electronic control strategy for the power train of hybrid electric vehicle,” in *Vehicle Electronics Conference, 1999. (IVEC ’99) Proceedings of the IEEE International*, 1999, pp. 383–386 vol.1.

[41] Z. Rahman, K. L. Butler, and M. Ehsani, “A comparison study between two parallel hybrid control concepts,” 03 2000.

[42] G. Paganelli, T. Guerra, S. Delprat, J. Santin, M. Delhom, and E. Combes, “Simulation and assessment of power control strategies for a parallel hybrid car,” *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 214, no. 7, pp. 705–717, 2000.

[43] A. Brahma, Y. Guezennec, and G. Rizzoni, “Optimal energy management in series hybrid electric vehicles,” in *Proceedings of the American Control Conference*, Chicago, IL, Sep 2000, pp. 60–64.

[44] S. Delprat, T. Guerra, and J. Rimaux, “Optimal control of a parallel powertrain: from global optimization to real time control strategy,” in *Vehicular Technology Conference, 2002. VTC Spring 2002. IEEE 55th*, vol. 4, Birmingham, AL, May 6–9 2002, pp. 2082–2088.

[45] I. Kolmanovsky, I. Siverguina, and B. Lygoe, “Optimization of powertrain operating policy for feasibility assessment and calibration: stochastic dynamic programming approach,” in *American Control Conference, 2002. Proceedings of the 2002*, vol. 2, 2002, pp. 1425–1430 vol.2.

[46] C. Lin, H. Peng, and J. Grizzle, “A stochastic control strategy for hybrid electric vehicles,” in *Proceedings of American Control Conference, 2004*, 2004.

[47] E. D. Tate, J. W. Grizzle, and H. Peng, “Shortest path stochastic control for hybrid electric vehicles,” *International Journal of Robust and Nonlinear Control*, vol. 18, pp. 1409–1429, 2008.

[48] L. Johannesson, M. Åsbogård, and B. Egardt, "Assessing the potential of predictive control for hybrid vehicle powertrains using stochastic dynamic programming," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, pp. 71–83, 2007.

[49] S. Moura, H. Fathy, D. Callaway, and J. Stein, "A stochastic optimal control approach for power management in plug-in hybrid electric vehicles," *Control Systems Technology, IEEE Transactions on*, vol. 19, no. 3, pp. 545 –555, may 2011.

[50] D. Opila, X. Wang, R. McGee, R. Gillespie, J. Cook, and J. Grizzle, "An energy management controller to optimally trade off fuel economy and drivability for hybrid vehicles," *Control Systems Technology, IEEE Transactions on*, vol. 20, no. 6, pp. 1490–1505, 2012.

[51] M. Widmer, "Stochastic dynamic programming for buses in public transportation," Master's thesis, ETH Zürich, 2011.

[52] UITP, "Project sort - standardized on-road test cycles," 2000.

[53] M. Back, "Prdiktive antriebsregelung zum energieoptimalen betrieb von hybridfahrzeugen," Ph.D. dissertation, Universitt Karlsruhe, 2005.

[54] L. Johannesson and B. Egardt, "A novel algorithm for predictive control of parallel hybrid powertrains based on dynamic programming," in *Proceedings of the 5th IFAC Symposium on Advances in Automotive Control*, Monterey, CA, August 20–22 2007.

[55] A. Sciarretta, L. Guzzella, and M. Back, "A real-time optimal control strategy for parallel hybrid vehicles with on-board estimation of the control parameters," in *Proceedings of IFAC*, 2004.

[56] C. Musardo, G. Rizzoni, Y. Guezennec, and B. Staccia, "A-ECMS: An adaptive algorithm for hybrid electric vehicle energy management," *European Journal of Control*, vol. 11, no. 4–5, pp. 509–524, 2005.

[57] C.-C. Lin, S. Jeon, H. Peng, and J. Moo Lee, "Driving pattern recognition for control of hybrid electric trucks," *Vehicle System Dynamics*, vol. 42, no. 1-2, pp. 41–58, 2004.

[58] L. Johannesson, S. Pettersson, and B. Egardt, "Predictive energy management of a 4qt series-parallel hybrid electric bus," *Control Engineering Practice*, vol. 17, no. 12, pp. 1440 – 1453, 2009, special Section: The 2007 IFAC Symposium on Advances in Automotive Control.

[59] J. T. Kessels, M. W. Koot, P. P. van den Bosch, and D. B. Kok, "Online energy management for hybrid electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 6, pp. 3428–3440, 2008.

[60] D. Ambühl and L. Guzzella, "Predictive reference signal generator for hybrid electric vehicles," *Vehicular Technology, IEEE Transactions on*, vol. 58, no. 9, pp. 4730–4740, Nov. 2009.

[61] van Keulen, T., de Jager, B., Serrarens, A., and Steinbuch, M., "Optimal energy management in hybrid electric trucks using route information," *Oil Gas Sci. Technol. - Rev. IFP*, vol. 65, no. 1, pp. 103–113, 2010.

Curriculum Vitae

Personal Data

Name:	Philipp Valentin Elbert
Date of Birth:	June 22, 1982
Place of Birth:	Starnberg, Germany
Nationality:	German

Education

1990 – 1993:	Südstadt Grundschule, Göppingen
1993 – 2002:	Freihof Gymnasium, Göppingen
2002 – 2003:	Social Service at Rotes Kreuz Göppingen
2003 – 2006:	B.Sc. ETH Degree in Mechanical Engineering, ETH Zürich
2006 – 2007:	Visiting Scholar at University of Toronto, Canada
2006 – 2009:	M.Sc. ETH Degree in Mechanical Engineering, ETH Zürich
2009 – 2013:	Doctoral Student and Teaching Assistant at the Institute for Dynamic Systems and Control, D-MAVT, ETH Zürich

Journal Articles

- Elbert, Philipp; Ebbesen, Soren; Guzzella, Lino "Implementation of Dynamic Programming for n-Dimensional Optimal Control Problems with Final State Constraints", *IEEE, Transactions on Control Systems Technology*, Vol.21, No.3, pp.924-931, 2013.
- Ebbesen, Soren; Elbert, Philipp; Guzzella, Lino "Battery State-of-Health Perceptive Energy Management for Hybrid Electric Vehicles" *IEEE, Transactions on Vehicular Technology*, Vol.61, pp.2893-2900, 2012.
- Ebbesen, Soren; Elbert, Philipp; Guzzella, Lino "Engine Downsizing and Electric Hybridization Under Consideration of Cost and Drivability" *Oil & Gas Science and Technology – Rev. IFP Energies Nouvelles*, Vol.67, pp.109-116, 2012.
- Elbert, Philipp; Nüesch, Tobias; Ritter, Andreas; Murgovski, Nikolce; Guzzella, Lino "Optimal Energy Management for a Serial Hybrid Electric Bus via Convex Optimization", Accepted for publication in *IEEE, Transactions on Vehicular Technology*, 2014.

Conference Articles

- Elbert, Philipp; Onder, Christopher; Gisler, Hans-Jörg "Capacitors vs. Batteries in a Serial Hybrid Electric Bus" *IFAC Symposium Advances in Automotive Control (AAC)*, 2010, Munich, Germany.
- Gisler, Hans-Jörg; Elbert, Philipp "Capacitors vs. Batteries in a Serial Hybrid Electric Bus" *9th Symposium on Hybrid and Electric Vehicles, IAV*, 2012, Braunschweig, Germany.
- Elbert, Philipp; Ebbesen, Soren; Guzzella, Lino "Economic Viability of Battery Load-Leveling in Hybrid Electric Vehicles using Supercapacitors" *Int. Scientific Conference on Hybrid Electric Vehicles, RHEVE*, 2011, Paris, France.