Optimum Energy Management Strategies for an Electric Vehicle Integrated in an Intelligent Transport System

Master's thesis

Alexander Massoner, BSc.

Institute of Automation and Control

Graz University of Technology

In cooperation with

AVL List GmbH

Supervisors:

Ao. Univ.-Prof. Dipl.-Ing. Dr.techn. Anton Hofer

Emre Kural, MSc.

Stephen Jones, Ph.D.

Graz, December 2012

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....

date signature

Zusammenfassung

Die vorliegende Arbeit befasst sich mit der Entwicklung neuer Strategien zur Erhöhung der Reichweite von Elektrofahrzeugen. Es werden dabei verschiedene Ansätze zur Energieverbrauchsoptimierung verfolgt, welche sich über unterschiedliche Zeithorizonte erstrecken und einander ergänzen. Eine dynamische Drehmomentaufteilung zwischen zwei Synchronmaschinen optimiert den Gesamtwirkungsgrad des Fahrzeugs. In Kapitel 4 werden zwei Verfahren vorgestellt, um energieoptimale Geschwindigkeitstrajektorien zu berechnen, für die zahlreiche praktische Anwendungsfälle existieren. Das letzte Kapitel beschäftigt sich mit der Entwicklung eines Konzeptes für einen Routenplaner, welcher den spezifischen Energieverbrauch eines Fahrzeugs in die Berechnung der Route einbezieht. Die entwickelten Optimierungsmethoden werden anhand des Fahrzeugmodells zweier realer Prototypen demonstriert. Die Anfertigung dieser Masterarbeit erfolgte in enger Zusammenarbeit mit der AVL List GmbH und im Rahmen des europäischen Forschungsprojektes OpEneR.

Abstract

The development of new strategies in order to increase the range of fully electric vehicles by minimizing the energy consumption using on-board and off-board sources of information has been subject of this work. Various complementary approaches that cover significantly different time horizons are analyzed. A method is developed to optimally distribute the drive torque between two electric machines aiming at maximizing the overall efficiency of the vehicle. Chapter 4 elaborates two computational procedures to determine energy-optimal velocity trajectories that can be applied in a broad variety of use cases. The last chapter illustrates a concept study for an advanced route planning optimization that takes into account powertrain characteristics in order to consider the expected energy consumption when calculating an optimal route. All developed optimization methods are based on parameters of two existing vehicle prototypes. This master's thesis was written in close collaboration with AVL List GmbH and contributes to the European research project OpEneR.

Schlüsselwörter

Bellman-Ford Algorithmus, B-splines, Elektrofahrzeug, OpEneR, dynamische Drehmomentverteilung, dynamische Programmierung, Parameteroptimierung, Routenplaner, Fahrroutenoptimierung, energieoptimale Geschwindigkeitstrajektorien, Energieoptimierung.

Keywords

Bellman-Ford algorithm, B-splines, Electric vehicle, OpEneR, dynamic torque distribution, torque split, dynamic programming, parameter optimization, route planner, route planning optimization, energy-optimal velocity trajectories, speed profile optimization, energy optimization.

Acknowledgment

This work has been supported by the OpEneR project, grant agreement number 285526, funded by the European Commission Seventh Framework Programme theme FP7-2011-ICT-GC.

Preface

An dieser Stelle möchte ich mich sehr herzlich bei allen Personen bedanken, die mich während meiner Arbeit unterstützt haben, im Besonderen bei Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. tit.Univ.-Prof. Anton Hofer seitens des Instituts für Regelungs- und Automatisierungstechnik an der Technischen Universität Graz, sowie bei meinen Betreuern bei AVL, Emre Kural, MSc. und Dr. Stephen Jones.

Ein ganz besonderer Dank gilt meinen Eltern, die mir stets mit Liebe zur Seite stehen.

Contents

List of Figures

List of Tables

List of Abbreviations

ACC: Adaptive Cruise Control

AVL: Anstalt für Verbrennungskraftmaschinen List

c2c: Car to car

c2i: Car to infrastructure

DC: Direct current

EC: European Commission

EM: Electric machine

ESP®hev: Electronic Stability Program for hybrid electric vehicles

FTP: EPA Federal Test Procedure

GPS: Global Positioning System

HMI: Human-machine interface

NEDC: New European Driving Cycle

OpEneR: Optimal Energy Consumption and Recovery based on a system network

rpm: Revolutions per minute

SISO: Single-Input Single-Output

SOC: State of charge

1 Introduction

Mobility plays a key role in every advanced economy and for individuals living in a modern society mobility has even become a basic need.[1] To a great extent individual mobility relies on passenger cars, which consume a large part of the world's primary energy carriers[2] but the imminent scarcity of these resources requires a shift away from fossil fuels. Especially electric vehicles have a promising future in the pursuit of this goal. However, there are still many obstacles that hinder electric vehicles to replace conventional cars or hybrids.

A crucial factor to customer acceptance is the generally low all-electric range, which is defined by the distance a vehicle can travel by only consuming the energy that is stored in its battery. In this thesis several energy-optimum control strategies are developed in MATLAB/Simulink and AVL CRUISE to raise the energy efficiency – and therefore increase the all-electric range – of two prototypes. The prototypes are fully electric vehicles adapted from the Peugeot 3008 HYbrid4. They each have two electric machines (EM) – one to power each axle – that facilitate regenerative braking. Details on specific vehicle data are elaborated in Appendix A.

This thesis is written in close collaboration with AVL List in Graz, Austria in line with the OpEneR (Optimal Energy consumption and Recovery based on a system network) project. OpEneR is a European research project launched in May 2011 and is part of the 7th Framework Programme (grant agreement n. 285526). The project goal is to develop an overall energy manager for electric vehicles that will significantly increase the energy efficiency and therefore the driving range as well as safety. OpEneR particularly emphasizes the development of energy saving strategies that combine data from car-to-car (c2c) systems, car-to-infrastructure (c2i) systems, GPS (Global Positioning System) data, nav-

[1] Cf. Back M. (2005): pp. 1.

[2] Cf. Guzzella L., Sciarretta A. (2007): pp. iii.

igation systems, cameras and/or radar to detect the vehicle's surrounding, etc.[3] The project consortium consists of the following institutions:

- Robert Bosch GmbH (project leader),
- Peugeot Citroën Automobiles S.A.,
- Robert Bosch Car Multimedia GmbH,
- AVL List GmbH,
- Centro Technológico de Automación de Galicia,
- FZI Forschungszentrum für Informatik.[4]

To explore the potential of the vast amount of data that will presumably be available in the near future several components are integrated to receive, combine and process data from a broad variety of sources to ensure a safe, highly efficient and comfortable driving experience. Those sources of information can be other vehicles that communicate with each other and share information about their current position, velocity etc. Real-time information can also be provided by modern road infrastructure that broadcasts information about speed limits, traffic flow, traffic light status etc. If this information is available a vehicle is integrated in an intelligent transport system. Furthermore vehicles are able to create lots of data on their own that can be utilized. For example, cameras and radar provide information about objects in the vehicle's surrounding. This approach is well suited to make traffic flow more smoothly.[5]

Compared to humans a major advantage of information systems is that they can gather and process much more data and are able to look ahead several kilometers along the road. This opens numerous so far unused possibilities to optimize driving. For this purpose an intuitive

[3] Cf. http://www.fp7-opener.eu, 4/27/2012.

[4] Cf. http://www.fp7-opener.eu/index.php/project/partners, 4/27/2012.

[5] Cf. http://www.fp7-opener.eu, 4/27/2012.

human-machine-interface (HMI) is installed in the vehicle to assist the driver.[6]

In this work it is assumed that some of the above mentioned information is available on-board and the vehicle is integrated in an intelligent transport system, respectively. A key distinguishing feature of the implemented strategies is that they cover a very different time horizon in order to be applicable in a broad variety of use cases that span from a duration of a few seconds to hours.

In the second chapter various components of the prototypes are explored for which all optimization strategies are designed. However, it should be emphasized that the developed algorithms can very easily be adapted to be applied on other fully electric vehicles with similar topology. Furthermore a vehicle model including all relevant components is derived.

The third chapter presents an algorithm that dynamically splits the torque between two electric machines depending on the driving condition. The purpose is to increase the combined efficiency of the electric machines.

The fourth chapter elaborates the implementation of energy-optimal velocity profiles. A velocity profile is a trajectory the vehicle speed should follow in a certain driving maneuver. The time horizon of such a maneuver is usually between 15 and 50 seconds.

In the last chapter a basic concept for a route planner is developed that – besides the distance and the journey time – takes into account the energy consumption. For this purpose 3D GPS maps, information about speed limits, real-time traffic conditions etc. are considered to be available on-board. The user can specify a tradeoff between energy consumption, travel time and traveled distance according to personal preferences.

---

[6] Cf. http://www.fp7-opener.eu/index.php/project/open-task-list.html, 26/9/2012.

2 Modeling of the vehicle

A suitable plant model is the foundation for any design process that aims at computing control strategies that can be successfully applied to a real dynamic system. For this reason the key elements of the vehicle model are briefly described in this chapter. The components of the prototypes that are relevant for the development of the optimization algorithms elaborated in the subsequent chapters are shown in Figure 2.1. The design process of the Torque Split Logic is explained in detail in chapter 3. Figure 2.1 also shows the underlying vehicle topology. Two electric machines that each power one axle are installed. Details on the electric machines and the ESP®hev (Electronic Stability Program for hybrid electric vehicles) designed by Bosch will be elaborated below in the corresponding subchapters.

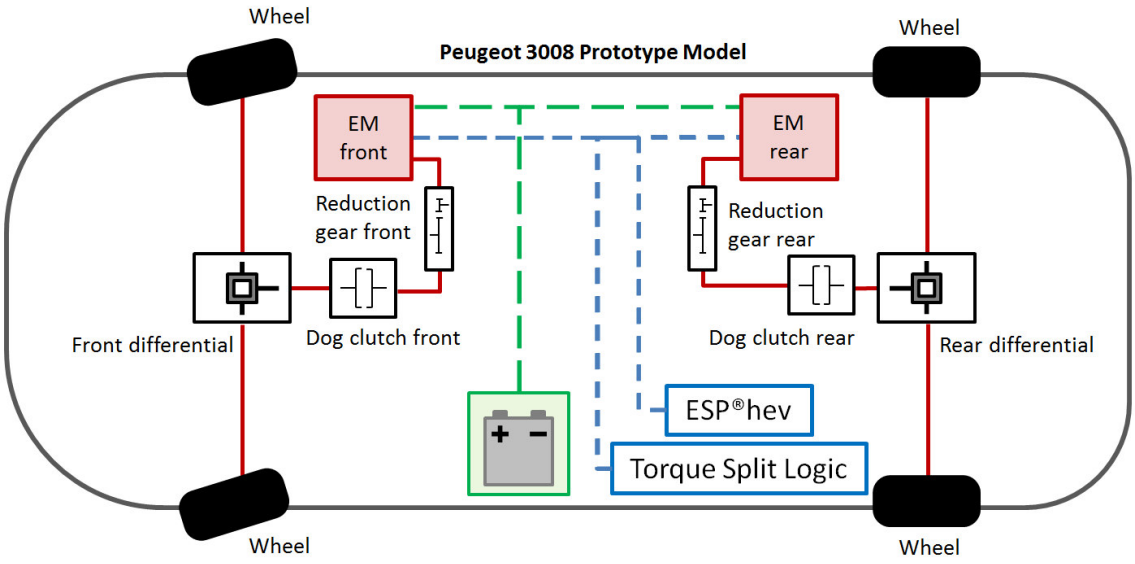

Figure 2.1: Considered components in the vehicle model.

The vehicle can – up to a certain limit specified by the ESP®hev – be slowed down by using the rear electric machine in generator mode to recuperate mechanical energy which is used to charge the on-board battery. Due to the fact that the energy dissipated in the mechanical brakes cannot be reused, the author assumes that the outcome of an energy-efficiency optimization will in no case result in the use of the mechanical brakes. Therefore, disk brakes are purposely not considered in the modeling process.

To simplify matters it is reasonable to neglect all lateral dynamics as well as pitch because both do not exert any significant influence to the energy consumption.[7] To describe the vehicle's longitudinal dynamics, the equations of motion with respect to specific vehicle data are derived below.

2.1 Longitudinal dynamics

Generally for realistic modeling of the longitudinal dynamics a single-track model is required to represent breaking and acceleration characteristics, which have a substantial influence on the stability of vehicles.[8] However, stability is not considered in this work because in any unstable driving situation the energy optimization becomes completely pointless. Besides, the electric machine torque can be limited in every developed algorithm to support stability. It is therefore convenient to neglect all stability concerns in the context of energy-optimization and use a single-wheel model. The equation of motion can be written according to Newton's second law as

$$I_{veh}\dot{\omega}_{wheel} = M_{PT} - M_{res}, \tag{2.1}$$

where

I_{veh} is the total vehicle inertia,
ω_{wheel} is the angular velocity of the wheel,
M_{PT} is the overall torque of the powertrain at the wheel and
M_{res} is the driving resistance at the wheel.

Note that equation (2.1) does not explicitly contain a term for brake torque. However, brake torque can be applied through the rear electric machine and is therefore part of M_{PT}. By convention propelling torques

[7] Cf. Back M. (2005): pp. 47.

[8] Cf. Grießler L. (2011): pp. 4.

are labeled a positive sign, whereas resistance torques have negative signs.[9]

With respect to $v = \omega_{wheel} r_{wheel}$ where v is the vehicle speed and r_{wheel} is the dynamic rolling radius (the effective tire radius if the vehicle is in motion[10]) of the wheels, equation (2.1) can be rephrased to

$$\dot{v} = \frac{r_{wheel}}{I_{veh}} (M_{PT} - M_{res}). \tag{2.2}$$

The total kinetic energy of the vehicle is given by

$$\frac{1}{2} I_{veh} \omega_{wheel}^2 = \frac{1}{2} m v^2 + 4 \frac{1}{2} I_{wheel} \omega_{wheel}^2 + \frac{1}{2} I_{PT} \omega_{wheel}^2, \tag{2.3}$$

where I_{PT} and I_{wheel} denote the inertia of the powertrain and the wheel inertia, respectively and m is the vehicle mass. Hence, with the relation $v = \omega_{wheel} r_{wheel}$ the vehicle inertia can be stated as[11]

$$I_{veh} = m r_{wheel}^2 + 4 I_{wheel} + I_{PT}. \tag{2.4}$$

2.2 Powertrain model

According to Figure 2.1 the powertrain model includes the electric machines, the transmissions, the dog clutches, the differentials and the wheels. It is asserted that the dog clutches are always closed and do not slip in normal driving, including pulling off from standstill. Torsion of the drive shaft is also neglected. Therefore, a quasi-static approach can be taken to model the powertrain. The transmission ratio is constant and defined as

$$i_{tr} = \frac{\omega_{EM}}{\omega_{wheel}}, \tag{2.5}$$

where ω_{EM} is the front and rear electric machine speed.

---

[9] Cf. Back M. (2005): pp. 47-48.

[10] Cf. Kiencke U., Nielsen L. (2000): pp. 249.

[11] Cf. Grießler L. (2011): pp. 3-4.

For simplicity the efficiencies of the transmission and the differential are regarded as constants and referred to as η_{tr} and η_{diff}, respectively. Note that both the front and the rear axle are constructed identically. Therefore the relation between the total power provided by both electric machines P_{2EM} and the total driving power at the wheels is given by

$$P_{2EM} = \omega_{EM} M_{2EM} = \frac{1}{\eta_{tr}\eta_{diff}} \omega_{wheel} M_{PT} = \frac{1}{\eta_{tr}\eta_{diff}} \omega_{EM} \frac{1}{i_{tr}} M_{PT}, \tag{2.6}$$

where M_{2EM} is the torque of both electric machines. It is now evident that the relation between the torque provided by the electric machines and the torque at the wheels is

$$M_{2EM} = \frac{1}{i_{tr}\eta_{tr}\eta_{diff}} M_{PT}. \tag{2.7}$$

Neglecting the inertia of the dog clutch the rotational energy of the powertrain is given by

$$\frac{1}{2} I_{PT} \omega_{wheel}^2 = \frac{1}{2} \left[\left(2I_{EM} + I_{tr,EM}\right) \omega_{EM}^2 + \left(I_{tr,wheel} + 2I_{diff}\right) \omega_{wheel}^2 \right], \tag{2.8}$$

where

I_{EM} is the inertia of one electric machine,

$I_{tr,EM}$ is the transmission inertia on the EM side,

$I_{tr,wheel}$ is the transmission inertia on the wheel side and

I_{diff} is the inertia of one differential.

By substituting ω_{EM} according to equation (2.5) the inertia of the whole powertrain can be determined by

$$I_{PT} = \left(2I_{EM} + I_{tr,EM}\right) i_{tr}^2 + I_{tr,wheel} + 2I_{diff}. \tag{2.9}$$

2.3 Driving resistance

The rolling resistance, the air resistance as well as the climbing resistance are represented by

$$M_{res} = (a_0 + a_2 v|v| + mg \sin\varphi) r_{wheel}, \tag{2.10}$$

where g is the gravitational acceleration, φ is the angle of inclination in radian and $a_0, a_1 \in \mathbb{R}$ are constants whose values were provided by Peugeot. Both constants were determined in coast-down tests on the road. Note that the term $v|v|$ will be replaced by v^2 in the equations below because driving backwards is not considered.

2.4 Electric machine efficiency model

To compute the overall energy consumption of the vehicle in any driving condition the electric machine efficiency is used. The efficiency is determined with the aid of a Simulink model using a C MEX-file S-function provided by the Robert Bosch GmbH. The model only represents the steady state behavior of the efficiency of one electric machine. All relevant inputs and the output of the model are shown in Figure 2.2.

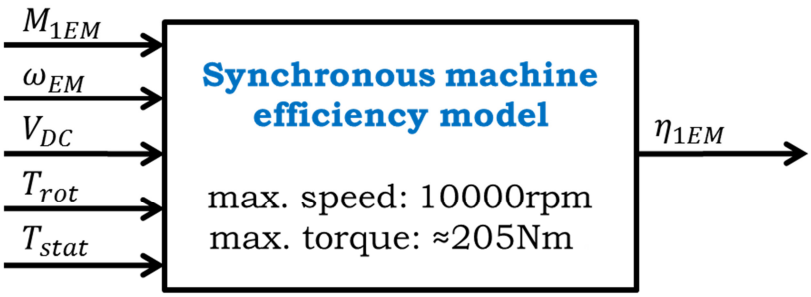

Figure 2.2: Electric machine efficiency model.

M_{1EM} denotes the electric machine torque (note the distinction between M_{1EM} and the combined torque of both EM M_{2EM}). V_{DC} is the DC (direct current) voltage at the clamps of the inverter of the electric machine and is provided by the battery. T_{rot} and T_{stat} refer to the rotor and stator temperature.

The electric machine efficiency is labeled η_{1EM}. Note that the efficiency can be considered as a function with five arguments. Hence, $\eta_{1EM} = \eta_{1EM}(M_{1EM}, \omega, V_{DC}, T_{rot}, T_{stat})$.

Optimizing the efficiency contributes substantially to reducing the overall energy consumption of the vehicle. Chapter 3 will purely focus on optimizing the combined efficiency of both electric machines.

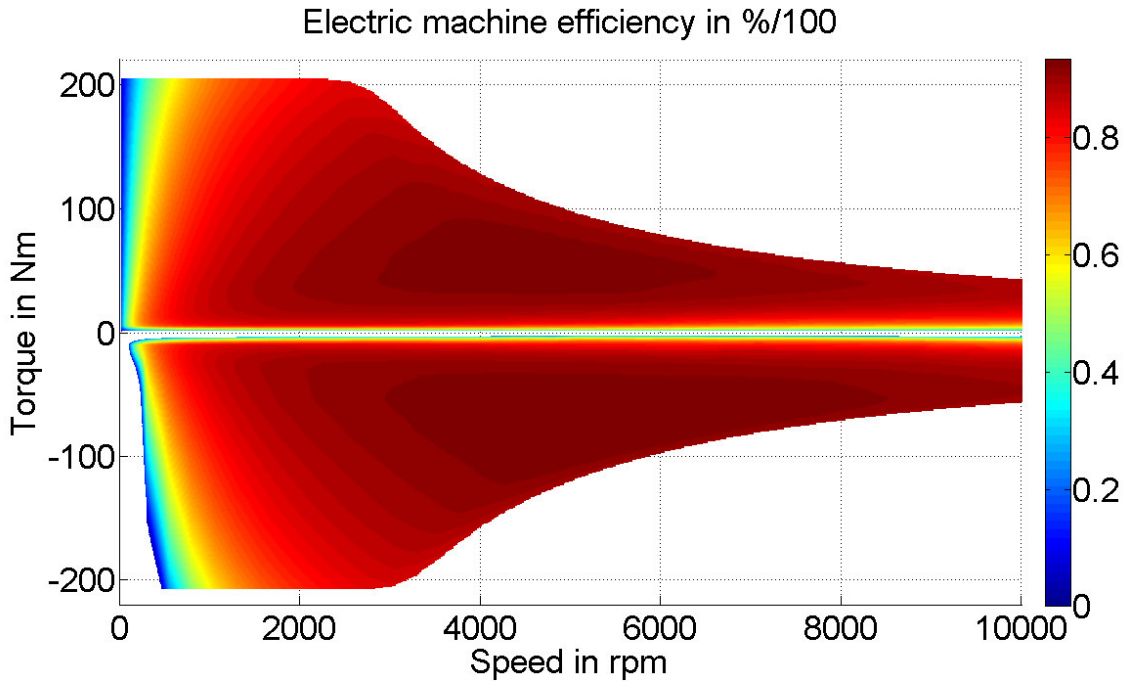

Figure 2.3: Efficiency of one electric machine, T_{rotor} = T_{stator} = 60°C, V_{DC} = 305V.

Let P_{el} be the electrical power one electric machine drains from the battery and let $P_{1EM} = \omega M_{1EM}$ be the mechanical power the electric machine provides so that the efficiency is defined as

$$\eta_{1EM} = \begin{cases} \frac{P_{1EM}}{P_{el}} \ldots P_{1EM},\ P_{el} > 0 \\ \frac{P_{el}}{P_{1EM}} \ldots P_{1EM},\ P_{el} < 0 \\ NaN \ \ldots otherwise, \end{cases} \tag{2.11}$$

where NaN stands for not a number. It is useful to use this notation because in the simulation environment any numerical value for η_{1EM} other than $0 \leq \eta_{1EM} \leq 1$ must be avoided. For $T_{rotor} = T_{stator} = 60℃$ and $V_{DC} = 305V$ Figure 2.3 shows the characteristics of η_{1EM} for both motor and generator mode. The white area contains any infeasible operation region as well as any points where according to equation (2.11) the efficiency is defined as NaN. In general, the efficiency decreases at higher

temperatures and vice versa. Some basic investigations on this phenomenon will be given in chapter 3.2.

2.5 ESP®hev in the prototypes

The ESP®hev enables regenerative breaking and has been specifically developed for hybrid and electric vehicles.[12] At the time the simulations for this work were conducted it allowed the test vehicle to purely regenerative decelerate up to approximately 1.25 m/s² by using only the rear electric machine. Energy recuperation with the front EM is not possible. At higher deceleration values regenerative breaking is mixed with hydraulic brake torque so as to the recuperation ability diminishes continuously until it reaches zero at a deceleration value of -3m/s² (see Figure 2.4).

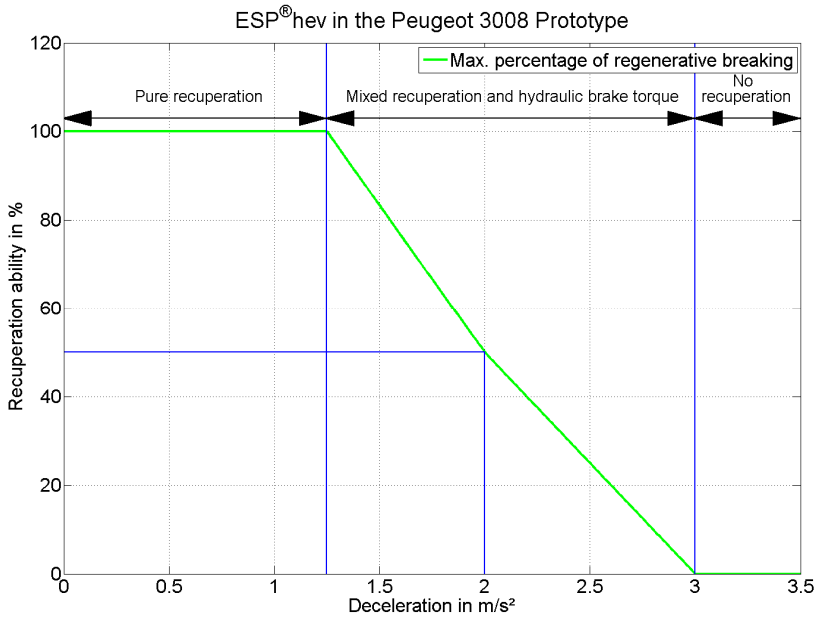

Figure 2.4: ESP®hev regenerative braking characteristics.

[12] Cf. Robert Bosch GmbH: http://www.bosch-presse.de/presseforum/details.htm?txtID= 5291, 7/18/2012.

2.6 State space representation

A state space model of the prototypes is given by

$$\dot{x} = f(x,u), \quad x(0) = x_0, \tag{2.12}$$

where $x = v$ is the only state variable, $u = M_{2EM}$ is the control input and x_0 is the initial condition. The vehicle speed v is easily measurable and represents the system output y. The state space model can therefore be stated as

$$\dot{v} = \frac{r_{wheel}}{I_{veh}}\left(i_{tr}\eta_{tr}\eta_{diff}M_{2EM} - (a_0 + a_2v^2 + mg\sin\varphi)r_{wheel}\right),$$
$$y = v, \quad v(0) = v_0. \tag{2.13}$$

2.7 Flatness

The solving method to the optimization problem in chapter 4.2 relies on a system property called flatness. This section gives a brief overview of flat systems and it is shown that the state space model of the prototypes features this property. For details the reader is referred to technical literature (e.g. Zeitz M. (2010), Gausch F. (2010), Grießler L. (2011)).

In general, flatness is a system property of dynamic systems. For linear systems flatness and controllability are interchangeable and therefore equivalent. For nonlinear systems flatness generalizes the property of controllability. For $dim(\boldsymbol{y}) = dim(\boldsymbol{u})$ the system

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{2.14}$$

is flat if it has at least one (possibly fictitious) flat output $\boldsymbol{y}$. The state vector $\boldsymbol{x}$ as well as all control inputs ($\boldsymbol{u}$) can be expressed explicitly by the flat output and a finite number of its derivatives:

$$\boldsymbol{x} = \boldsymbol{g_1}\left(\boldsymbol{y}, \dot{\boldsymbol{y}}, \dots, \boldsymbol{y}^{(n-1)}\right),$$
$$\boldsymbol{u} = \boldsymbol{g_2}\left(\boldsymbol{y}, \dot{\boldsymbol{y}}, \dots, \boldsymbol{y}^{(n)}\right). \tag{2.15}$$

Even if these conditions are just locally fulfilled the system is called flat.

Flatness is a very useful system property if trajectory planning is desired. To illustrate this, consider a Single-Input Single-Output (SISO) system where the system output must follow a specific trajectory $r(\boldsymbol{p}, t)$ where $\boldsymbol{p}^T = [p_1 \quad p_2 \quad p_3]$ is a parameter vector. Due to $y \stackrel{!}{=} r(\boldsymbol{p}, t)$ the system output can be parameterized as $y = y(\boldsymbol{p}, t)$. In this case flatness implies that not just the system output but the state vector as well as the control input depend on only $\boldsymbol{p}$ and t. In chapter 4.2 this property will be utilized to compute control inputs based on a set of parameters that specify a velocity trajectory.[13] At this point it is left to show that the vehicle model is indeed flat.

With $y = v$ equation (2.13) can be rearranged to

$$M_{2EM}(y, \dot{y}) = \frac{1}{i_{tr}\eta_{tr}\eta_{diff}}\left(\dot{y}\frac{I_{veh}}{r_{wheel}} + (a_0 + a_2 y^2 + mg\sin\varphi)r_{wheel}\right). \quad (2.16)$$

It is now evident that both v and M_{2EM} depend only on the system output and its first derivative. Therefore, the system is flat.

[13] Cf. Zeitz M. (2010): pp. 1, 3.

Cf. Gausch F. (2010): lecture notes.

Cf. Grießler L. (2011): pp. 40-41.

3 Torque split

Up to half load a vehicle that has two identical electric machines can drive by using either just one EM or the torque can be allocated arbitrarily between both EM. If a higher drive torque is required, both electric motors have to be used but as long as the demand torque is lower than the maximum torque of both electric machines there is still an infinite number of possibilities to unequally distribute the torque. However, without consideration of the electric machine efficiency characteristics no reasonable predication can be made about the optimal torque split. A strategy without any computational or implementation effort is to just even distribute the torque between both EM. This method also covers full load operation. It will serve as a benchmark for the optimization algorithm below.

The idea that is followed in this chapter is to reduce the power consumption by applying a dynamic distribution of the drive torque between both electric machines. The goal is therefore to always operate the electric machines at their combined maximum efficiency at any given torque demand by the driver in any driving condition. The combined maximum efficiency also depends on the temperature of the electric machines as well as the battery voltage. A detailed description will be given in the subchapters below. All algorithms are developed in MATLAB/Simulink using the vehicle model elaborated above. The effect on the overall energy consumption is then investigated by evaluating the torque split strategy in various driving cycles and driving maneuvers using a detailed model of the prototypes in AVL CRUISE.

3.1 Electric drive efficiency

Similar to equation (2.11) the combined efficiency of both EM is generally defined by

$$\eta_{2EM} = \begin{cases} \dfrac{P_{1EM,f} + P_{1EM,r}}{P_{el,f} + P_{el,r}} \ldots P_{1EM,f} + P_{1EM,r}, P_{el,f} + P_{el,r} > 0 \\ \dfrac{P_{el,f} + P_{el,r}}{P_{1EM,f} + P_{1EM,r}} \ldots P_{1EM,f} + P_{1EM,r}, P_{el,f} + P_{el,r} < 0 \\ NaN \ \ldots otherwise, \end{cases} \tag{3.1}$$

where

$P_{1EM,f}$ and $P_{1EM,r}$ denote the mechanical power of the front and the rear electric machine and

$P_{el,f}$ and $P_{el,r}$ is the electric power consumed or regenerated by the front and the rear electric machine, respectively.

However, due to the ESP®hev the front EM may not be used to recuperate energy. Therefore, equation (3.1) is simplified to

$$\eta_{2EM} = \begin{cases} \dfrac{P_{1EM,f} + P_{1EM,r}}{P_{el,f} + P_{el,r}} \ldots P_{1EM,f} + P_{1EM,r}, P_{el,f} + P_{el,r} > 0 \\ \dfrac{P_{el,r}}{P_{1EM,r}} \ldots P_{1EM,r}, P_{el,r} < 0 \\ NaN \ \ldots otherwise, \end{cases} \tag{3.2}$$

The torque split factor states what percentage of the total EM drive torque is provided by the front electric machine:

$$\mathrm{u}_f = \frac{M_{1EM,f}}{M_{1EM,f} + M_{1EM,r}}, \tag{3.3}$$

where

$M_{1EM,f} = \frac{P_{1EM,f}}{\omega_{EM}}$ is the torque provided by the front EM and

$M_{1EM,r} = \frac{P_{1EM,r}}{\omega_{EM}}$ is the rear torque.

Note that $0 \leq \mathrm{u}_f \leq 1$. The total electrical power is given by $P_{el,tot} = P_{el,f} + P_{el,r}$ and the total mechanical power is given by $M_{2EM} = M_{1EM,f} + M_{1EM,r}$. Figure 3.1 illustrates the connection between the overall mechanical and electric power of both electric machines for $P_{2EM}, P_{el,tot} > 0$ with regard to the torque split factor.

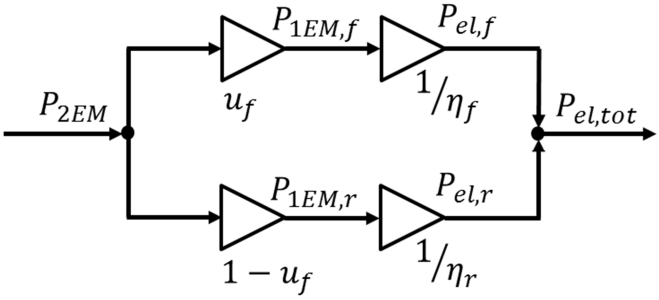

Figure 3.1: Torque split between front and rear EM.

It is now apparent that for $P_{2EM}, P_{el,tot} > 0$ the total electrical power can be calculated with

$$P_{el,tot} = \frac{u_f P_{2EM}}{\eta_f} + \frac{(1-u_f)P_{2EM}}{\eta_r}, \quad P_{2EM}, P_{el,tot} > 0. \tag{3.4}$$

Rearranging this equation gives

$$\frac{P_{2EM}}{P_{el,tot}} = \eta_{2EM} = \frac{1}{\frac{u_f}{\eta_f} + \frac{(1-u_f)}{\eta_r}} = \frac{\eta_f \eta_r}{u_f \eta_r + (1-u_f)\eta_f}, \quad P_{2EM}, P_{el,tot} > 0. \tag{3.5}$$

It is important to note that equation (3.5) shows that $\eta_{2EM} = g(u_f)$, where g is a nonlinear function of the torque split factor u_f. Therefore, an optimization of u_f results in the maximization of g. This is equivalent to reducing the energy consumption in any given driving condition where $P_{2EM}, P_{el,tot} > 0$.

Remark: If it was possible to use the front electric machine for recuperation as well, another version of equation (3.5) could be derived similarly for $P_{2EM}, P_{el,tot} < 0$ based again on Figure 3.1 but by substituting $1/\eta_f$ and $1/\eta_r$ for their reciprocal values. A new function $\tilde{g}(u_f)$ could be determined for recuperation mode.

Various optimization methods could be applied to find the maximum of g. The fact that u_f lies between zero and one simplifies matters considerably. In a special case where the supply voltage as well as the rotor and stator temperatures are equal for both EM this interval can be narrowed even further. Consider for example two identical EM under equal conditions: a torque split factor of $u_f = \alpha$ will give the exact same results as $u_f = 1 - \alpha$ $(0 \leq \alpha \leq 1)$. It is therefore evident that – with respect to the

conditions mentioned above – it is sufficient to consider only values of u_f in the interval $[0, 0.5]$.

An optimization of the torque split factor can therefore be carried out as follows. First of all $n \in \mathbb{Z}^+$ different values for u_f are chosen: $u_f = \{u_{f,1}, u_{f,2}, \dots, u_{f,n}\}$. Then simulations with all $u_{f,i}$ $(i = 1 \dots n)$ are run and the efficiency values are stored in the vector $\boldsymbol{\theta}^T = [\theta_1, \dots, \theta_n]$. If n is chosen adequately a sufficiently accurate approximation of the optimal torque split factor can be obtained with:

$$j = \min\{i: \theta_i = \|\boldsymbol{\theta}\|_\infty\}.$$
$$u_{f,opt} = u_{f,j}. \tag{3.6}$$

This procedure is systematically repeated for all feasible combinations of torque and speed and the optimal torque split values are stored in a lookup table. Temperature differences between the electric machines can be considered as well. Figure 3.2 shows $u_{f,opt}(\omega, \mathrm{M}_{2EM})$ for $V_{DC} = 305V$, $T_{rot} = T_{stat} = 60°\mathrm{C}$ (for both electric machines) and $n = 11$. The corresponding values for $u_{f,i}$ are $\{0, 0.05, 0.1, \dots, 0.5\}$.

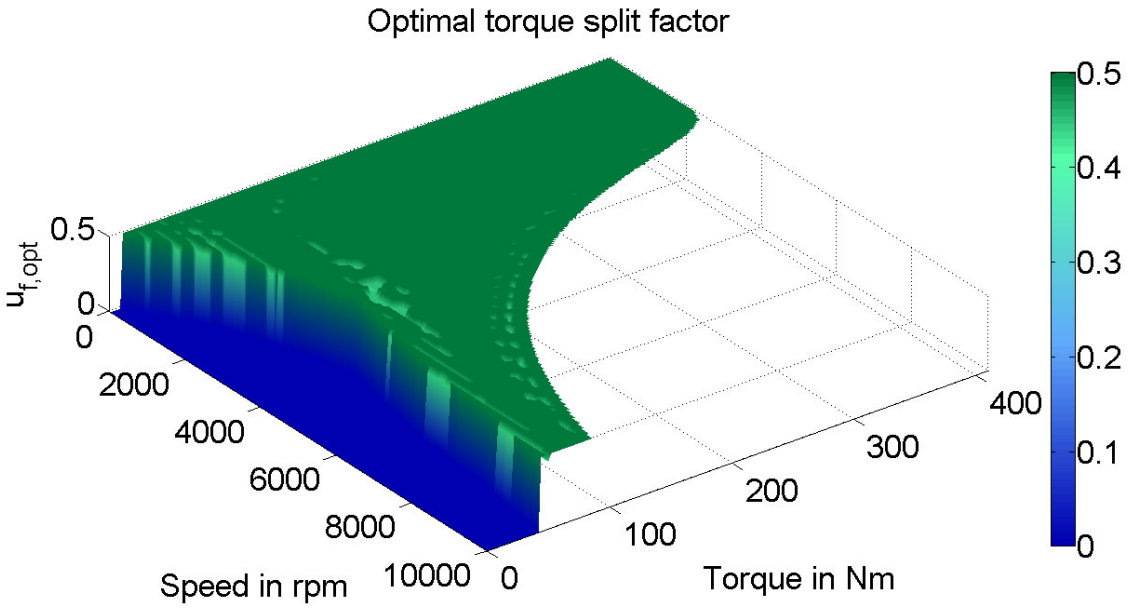

Figure 3.2: Lookup table for $u_{f,opt}$: $V_{DC} = 305V$, $T_{rot} = T_{stat} = 60°C$, $n = 11$.

Note that $u_{f,opt}$ is zero if the torque demand is low. This means that only one electric machine should be used. In higher load conditions the torque is mostly to be distributed equally between both electric ma-

chines. It is evident that only two modes are sufficient to optimally operate the vehicle in almost all operating points. Considering that the described torque split may be implemented in one of the prototypes it is desired to keep the implementation effort as low as possible as long as the tradeoff between improving the energy-efficiency and reducing complexity is reasonable. Thus, the question arises whether the lookup table for $u_{f,opt}$ could be simplified, i.e. reduced to fewer values. A first step is to investigate the impact of the simplification of $u_{f,opt}$ to just two values.

Based on Figure 3.2 a new variable $M_{threshold}$ is defined which shall represent a torque threshold. For any speed the threshold equates to the minimum torque value where $u_{f,opt}$ is not zero. Expressed mathematically the threshold is defined by

$$M_{threshold} = \min\{\mathrm{M}_{2EM}: u_{f,opt}(\omega, \mathrm{M}_{2EM}) > 0\}. \tag{3.7}$$

Now a simplified torque split lookup table can be defined as

$$u_{f,subopt} = \begin{cases} 0 \ldots & M_{2EM} < M_{threshold} \\ 0.5 \ldots & M_{2EM} \geq M_{threshold}. \end{cases} \tag{3.8}$$

Figure 3.3 shows $M_{threshold}$ for various supply voltage levels. Once more it is assumed that the supply voltages as well as the rotor and stator temperatures of the front and rear electric machine are equal.

Apparently, the threshold levels differ at higher speeds. Up to approximately one third of the maximum speed there is no significant voltage-dependency (for relevant values of the supply voltage). If implemented in the prototype, $u_{f,subopt}$ would not only require less implementation effort but also significantly less memory than $u_{f,opt}$. Whether the loss of energy-efficiency is worth it will be investigated in the next subchapter.

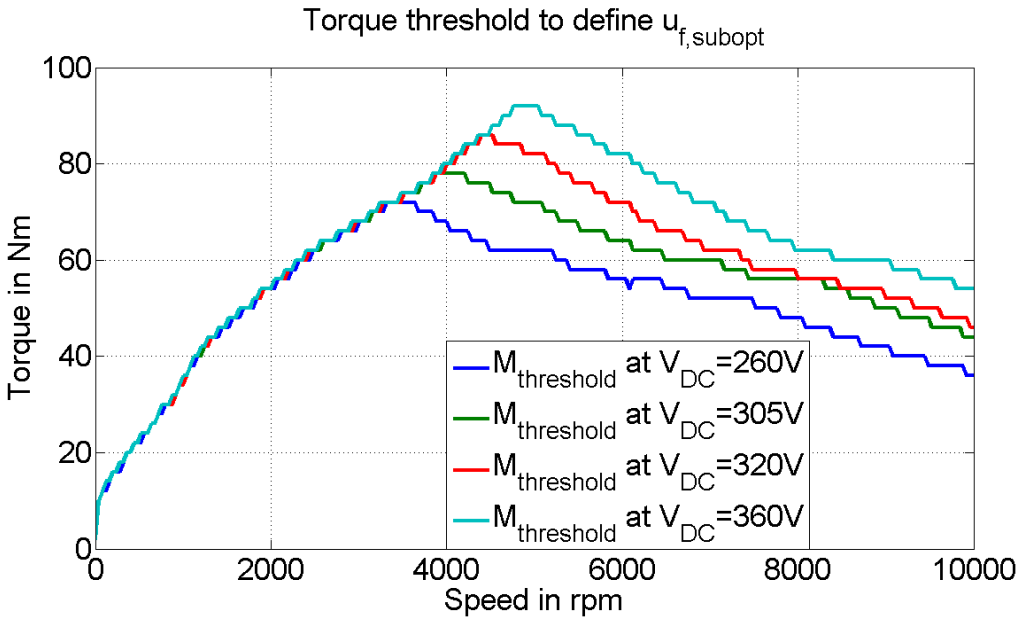

Figure 3.3: $M_{threshold}$ at different supply voltage levels.

3.2 Implementation and results

So far it has been shown that two different approximations to an optimal torque distribution can be determined. However, no investigation has yet been made whether the potential of reducing the energy consumption justifies the implementation effort. Thus, the impact of applying $u_{f,opt}$ and $u_{f,subopt}$ on the energy consumption is evaluated under various driving conditions. For this purpose a detailed model of the prototype in AVL CRUISE is used. Note that neither the optimal nor the suboptimal torque split are calculated in AVL CRUISE but implemented in the model after being determined in MATLAB as described in chapter 3.1. Note also that the AVL CRUISE model considerably differs from the one that is described in the second chapter because in contrast to the one-wheel model, all four wheels are considered. Also, the mechanical brakes can be used if pure regenerative breaking cannot provide enough break torque to carry out a specified driving maneuver. In addition a human driver model is included. Hence, it is assumed that the effect of the torque splitting can be realistically analyzed and that based on the

results a substantiated decision can be made whether it is worth to pursue this approach.

To evaluate the benefit of a torque distribution a suitable and precise criterion and an evaluation method, respectively are indispensable. For this purpose the application of both $u_{f,opt}$ and $u_{f,subopt}$ is compared to a static torque distribution which – under all driving conditions – splits the torque equally between both EM ($u_f = 0.5$). New variables δ_{opt} and δ_{subopt} are defined which represent the amount of energy saving in % compared to a static equably distribution of the torque:

$$\delta_{opt} = \frac{E_{u_f=0.5} - E_{u_f,\mathrm{opt}}}{E_{u_f=0.5}} * 100\%,$$

$$\delta_{subopt} = \frac{E_{u_f=0.5} - E_{u_f,\mathrm{subopt}}}{E_{u_f=0.5}} * 100\%. \tag{3.9}$$

$E_{u_f=0.5}$ denotes the total amount of the prototype's consumed energy in a specific driving maneuver or a driving cycle if a static and even torque distribution is applied. Accordingly $E_{u_f,\mathrm{opt}}$ and $E_{u_f,\mathrm{subopt}}$ denote the energy consumption if $u_{f,opt}$ or $u_{f,\mathrm{sub}opt}$ is applied.

At the juncture this thesis is written the author has no access to any reliable data concerning the thermal behavior of the electric machines (e.g. the thermal capacity or the specification of the on-board cooling system). Therefore a constant electric machine temperature is assumed during the following driving maneuvers.

Table 3.1 illustrates the results for the New European Driving Cycle (NEDC) as well as the EPA Federal Test Procedure (FTP-75). In addition the energy saving while driving at various constant velocities is shown. In all simulations the initial state of charge (SOC) of the battery is 88% and $T_{stator} = T_{rotor} = 60℃$ for both electric machines.

Energy consumption improvement due to torque splitting						
-	**NEDC**	**FTP-75**	**30 km/h**	**50 km/h**	**90 km/h**	**120 km/h**
δ_{opt}	5.423%	3.626%	23.691%	17,559%	7,422%	5,504%
δ_{subopt}	5.418%	3.624%	23.691%	17,559%	7,422%	5,504%

Table 3.1: Reduction of the consumed energy by dynamically distributing the torque between the electric machines.

The results show that the performance of $u_{f,subopt}$ is nearly as good as the one of $u_{f,opt}$. Thus, answering the question from the previous section whether it is useful to simplify $u_{f,opt}$ to reduce the implementation effort can clearly be answered indisputably. Especially at low load conditions the improvement is significant. The reason why the improvement values are equal at the constant velocities shown in Table 3.1 is that under those conditions both $u_{f,opt}$ and $u_{f,subopt}$ are 0. In comparison the relative energy saving is rather low in the examined driving cycles because they include lots of acceleration phases with relatively high torque demands compared to driving at constant speed. In those operating regions the optimal torque split is – according to Figure 3.1 – 0.5 which obviously achieves no advantage compared to a static and even torque splitting.

As shown above the improvement for the NEDC is about 5.4%. The purpose of Figure 3.4 is to better explain the NEDC and to show in which operating regions one or both electric machines are used. The figure shows both the vehicle speed during the NEDC (blue curve) and the corresponding suboptimal torque split factor $u_{f,subopt}$ (green curve). Both electric machines are used for accelerating at low speeds. At higher speeds only one machine is in use. Note the connection to Figure 3.3 where (for low velocities) the torque threshold rises as the speed increases. Also, as stated above $\delta_{subopt} = 0$ if the vehicle travels at a constant velocity of 120km/h or less.

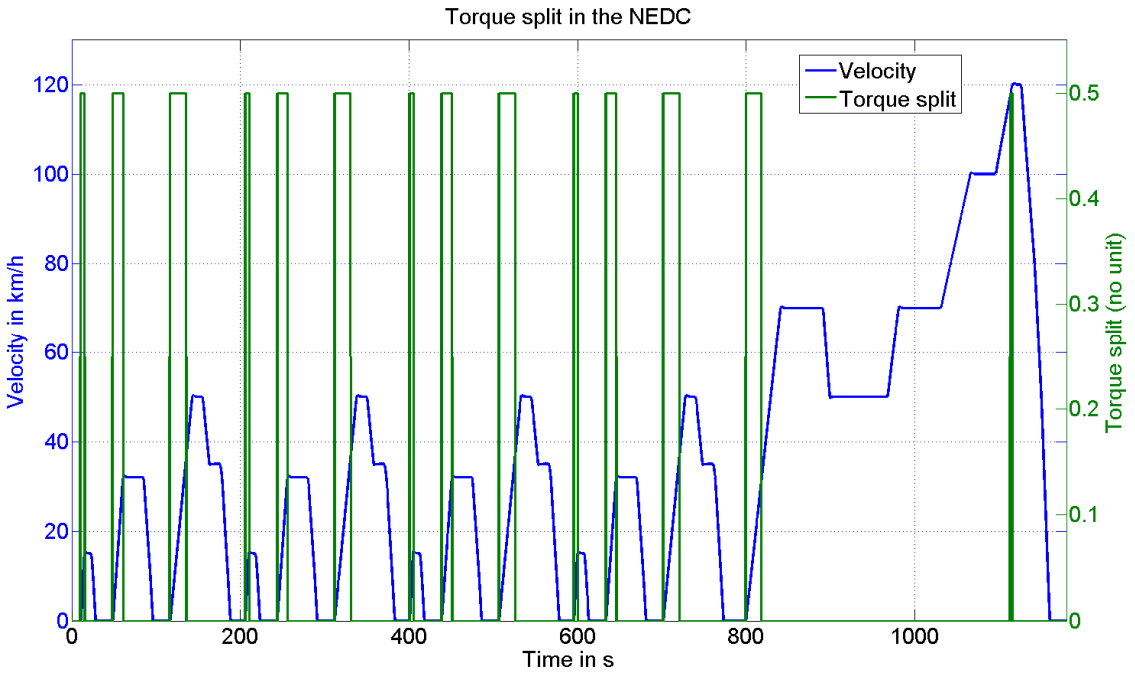

Figure 3.4: Suboptimal torque split during the NEDC.

A remark to the torque split simulations:

Although optimal torque split values can be determined for any temperature differences between the electric machines, so far the EM temperature has always been considered equal in the simulations conducted in AVL CRUISE. This simplification is indeed very useful because it allows obtaining simulation results for various driving scenarios without any knowledge of the thermal behavior of the electric machine and the cooling system. However, the question arises, whether this simplification renders the obtained results meaningless.

It can be shown in simulations that in general the efficiency of the electric machine decreases as the temperature increases. This is exemplified in Figure 3.5 for $V_{DC} = 305V$. The temperature refers to both T_{rot} and T_{stat}. For better comparability the torque is normalized to a percentage value of the maximum available torque at the correspondent temperature and driving speed.

In reality, whenever the torque is not distributed equally between the electric machines, they heat differently. The optimal torque distribution

changes accordingly. The effect on the overall energy-efficiency improvement is yet to be determined in further simulations or through measurements in the prototypes.

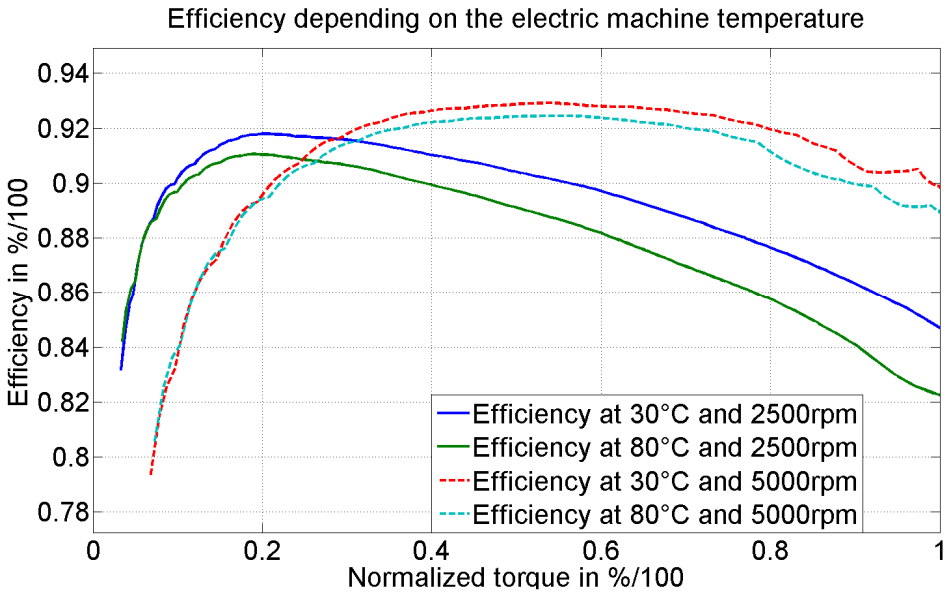

Figure 3.5: Electric machine efficiency depending on temperature.

Nevertheless, in the opinion of the author the results above are indeed meaningful and justify the implementation effort of adding a dynamic torque distribution in the algorithms presented in chapters 4 and 5. The reasons for this conclusion are:

1. The high relative energy saving values in Table 3.1 for constant velocities (or low load conditions) where only one EM operates can be approximated by alternately switching the torque between the front and rear EM. Thereby the EM temperatures are roughly constant as it was assumed in the conducted simulations.
2. At least after a sufficiently long standstill the EM temperatures are temporarily roughly equal and in this lapse of time the results shown in Table 3.1 are significant.
3. Preliminary and basic considerations give no serious indications that if the thermal characteristics were added the energy savings would be significantly worse than stated in Table 3.1.
4. Regenerative breaking is limited to the use of the rear electric machine. Therefore, even with static and even torque splitting the

rear electric machine eventually becomes hotter than the front EM. In this case a dynamic torque distribution is able to react to the subsequent decline of the efficiency at the rear electric machine.

5. The idea of a dynamic torque distribution might be extended to including estimations of future load conditions. These estimations can be based on 3D GPS data and real-time traffic conditions received through c2c and c2i communication. Instead of optimizing the instantaneous efficiency, a tradeoff between current energy-saving and future efficiency gain is possible. This would lead to sophisticated thermal management strategies and significantly contribute to the development of an advanced energy manger as it is requested in the OpEneR project.

4 Velocity Trajectory Optimization

In everyday traffic situations a driver has to adapt the traveling speed in order to meet a speed limit, stop at a traffic light or simply maintain a safe distance to other vehicles. Up to a certain degree it is up to the driver whether his or her driving is rather aggressive or energy efficient and foresightful. Particularly regenerative breaking enables motorists to combine anticipatory driving with low energy consumption. However, when it comes down to considering the overall powertrain efficiency, recoverable breaking energy, road inclination, traffic signs as well as predicting other road users behavior all at once while trying to safely and energy-efficiently steer a vehicle, even a skilled driver might feel a little overburdened.

The aim of this chapter is to develop and apply algorithms that can calculate optimum control inputs for fully electric vehicles to adapt the vehicle speed in various traffic situations. Given a particular traffic situation, these optimum control inputs are equivalent to optimal velocity trajectories the vehicle must follow within a predefined period of time or over a certain distance. In any case safe and comfortable driving must be ensured by simultaneously maximizing energy efficiency.

An important aspect is that for certain driving maneuvers it is possible to compute optimal control inputs a priori. This is interesting because it is state of the art that (with some limitations) longitudinal control of the vehicle can be taken over completely by adaptive cruise control (ACC). Therefore, in case a vehicle is equipped with ACC, energy-optimal driving can be combined with the safety and comfort autonomous driving provides. In this case the optimized actuating variable can be directly applied as a specific torque set point at the electric machines. Otherwise, recommendations can be shown to the driver through a human-machine interface (HMI). Consequently, the motivation behind the application of optimum speed profiles is to provide improved energy-efficiency while considering both safety and driving comfort.

The first part of this chapter deals with an optimization method called dynamic programming (DP). Foremost, the basic theoretical concept will be explained. Subsequently, the implementation of a DP algorithm in MATLAB is described. Results of energy-optimum velocity profiles that also take into account some basic considerations about driving comfort are shown. Being able to compute globally optimal solutions, DP is well-suited for benchmarking with other optimization tools. In the second part of this chapter another algorithm is developed, applying a parameter optimization method, which utilizes the fact that the controlled system (vehicle) features a flat output as described in chapter 2.7. As almost everywhere, different methods provide diverse upsides and downsides. Although the second procedure does not ensure the solution to be the globally optimal one, additional boundary conditions can be considered. Furthermore, it creates innately smoother velocity profiles and therefore increased driving comfort.

Both algorithms explained in this chapter are designed for relatively short maneuvers that take up to one minute and 1000m, respectively (although, it is possible to extend these methods to compute significantly longer velocity profiles). Thus, for simplicity it is assumed that both the electric machine temperatures and the battery voltage do not significantly change during such a short period of time. For this reason, V_{DC}, T_{Stator} and T_{Stator} are set as constant and will be omitted (e.g. the electric machine efficiency $\eta_{2EM}(M, \omega)$ has just two arguments). Furthermore, suboptimal torque splitting as described in chapter 3.1 is included to both algorithms.

Still missing in the introductory part of this chapter is a rather precise mathematical formulation of the optimization process and a strict definition of velocity profiles. It is desired to change the vehicle velocity v from any initial speed $0 \leq v_0 \leq v_{max}$ to another speed $0 \leq v_N \leq v_{max}$. In some cases it is useful to specify a period of time $T = t_N - t_0$ in which the transition from v_0 to v_N is to be carried out. For instance, if a motorist who uses ACC sets a desired velocity v_N it must be assured that after a finite period of time T the vehicle will travel at that speed.

However, in many cases it is impossible to exactly determine an instant in time at which the vehicle will reach a speed limit, an object on the road, a curve etc. These events are usually specified by distance values. Therefore, lots of use cases exist where it is rather useful to ensure that the vehicle reaches v_N within a certain covered distance $D = d_N - d_0$ instead of a specific period of time T. For example, at an upcoming lowered speed limit the motorist must of course adapt the traveling speed to satisfy the condition $v(d_N) \leq v_N$.

In order to cover the mentioned use cases, two different sorts of boundary conditions are considered:

i) $v_0 = v(t = t_0)$, $v_N = v(t = t_N)$ and

ii) $v_0 = v(d = d_0)$, $v_N = v(d = d_N)$.

It is therefore useful to specify the velocity as a function of time $v = v(t)$ in case i and as a function of the traveled distance $v = v(d)$ in case ii (see Figure 4.1).

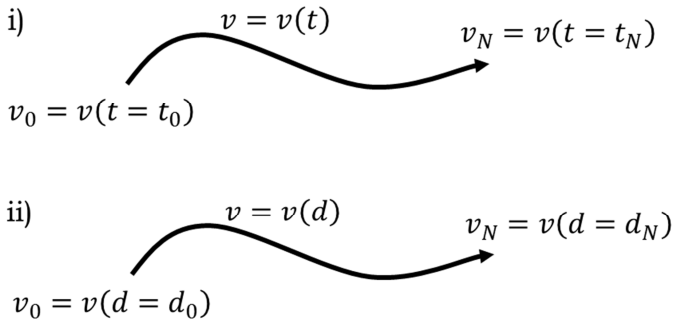

Figure 4.1: Vehicle velocity.

As it is quite impractical to always distinguish between $v(t)$ and $v(d)$ a new variable ξ is introduced to cover both cases (i and ii) by the notation $v = v(\xi)$, $v_0 = v(\xi_0)$ and $v_f = v(\xi_N)$. Now, the basic idea of energy-optimal speed profiles is to minimize the overall energy consumption J within the interval $[\xi_0, \xi_N]$, respecting the boundary conditions mentioned above as well as velocity-dependent acceleration limits and an upper and lower bound for $v(\xi)$. Therefore the optimization problem can be formulated as

$$\min_{v(\xi)}\{J\} = \min_{v(\xi)}\left\{\int_{\xi_0}^{\xi_N} P\left(v(\xi), \frac{\mathrm{d}v(\xi)}{\mathrm{d}\xi}\right)\mathrm{d}\xi\right\}$$

subject to:

$$v_{min} \leq v(\xi) \leq v_{max}$$
$$a_{min}\big(v(\xi)\big) \leq \frac{dv(\xi)}{d\xi} \leq a_{max}\big(v(\xi)\big), \qquad (4.1)$$

where P is the power consumption and a_{min} and a_{max} are limits for the vehicle acceleration due to limited electric machine power, safety or comfort reasons.

4.1 Velocity trajectory optimization with dynamic programming

In this thesis dynamic programming is used to compute optimized control inputs for a nonlinear dynamic system. However, in addition to optimal control theory, there is a broad variety of problems dynamic programming algorithms can be applied on. To explain the basic idea of DP, a situation is considered, where decisions are made in several stages. The objective is to particularly make those decisions that result in a desired outcome. In other (mathematical) words, it is desired to always make decisions that minimize a certain cost. A crucial factor in such situations is that even optimal choices in every isolated decision are unlikely to result in a satisfying final outcome. That is, because one must always consider future decisions as well in order to balance reducing current costs with receiving the chance to lower future costs. And this, fortunately, is a task DP can accomplish. At each stage, a decision is evaluated by the sum of the present cost and the overall future cost.[14]

[14] Cf. Bertsekas D., (2005): pp. 2.

Introductory example: a staged decision problem:

To illustrate the basic concept of dynamic programming, consider a multistage decision process as depicted in Figure 4.2. The first decision made at A results in the cost J_{AB}. The next decision can either lead to C, D or E incurring a cost of J_{BC}, J_{BD} or J_{BE}. From an isolated point of view in B it makes sense to proceed to C or D because $J_{BC} < J_{BE}$ and $J_{BD} < J_{BE}$, respectively. However, since neither C nor D lie on the optimal path (A-B-E) due to $J_{BC} + J_{CE} > J_{BE}$ and $J_{BD} + J_{DE} > J_{BE}$, proceeding to C or D is definitely not an optimal choice, considering the bigger picture.[15] After these preliminary considerations, it is now shown how this problem can be solved by means of dynamic programming.

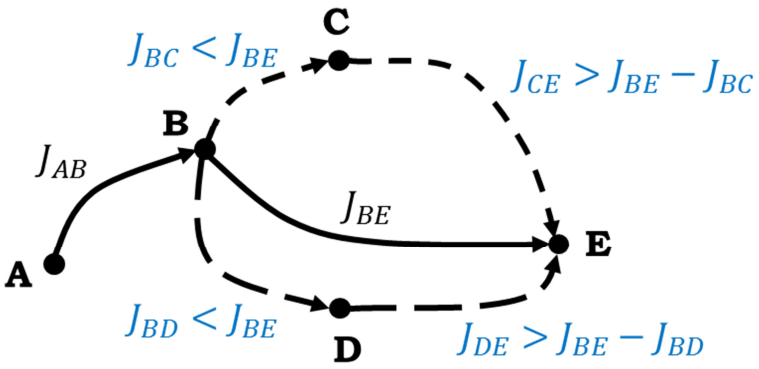

Figure 4.2: A multistage decision process.
Adapted from Kirk D. (2004): pp. 54.

A characteristic of DP algorithms is that they solve problems backwards. Therefore, the destination E is considered first, which can be accessed via B, C and D. The (optimal) cost to reach E from C can be written as $J_C^* = J_{CE}$ and is stored. Accordingly, the cost from D is $J_D^* = J_{DE}$, which is again stored. The optimal cost from B is now given by $J_B^* = min\{J_{BE}, (J_{BC} + J_C^*), (J_{BD} + J_D^*)\} = J_{BE}$. It is important to note, that for B two pieces of information must be stored:

i) The minimum cost to reach E is $J_B^* = J_{BE}$.
ii) The optimal decision at B is to proceed directly to E.

[15] Cf. Kirk D. (2004): pp. 54.

Lastly, B can be accessed via A. The optimal cost from A to the destination E is $J_A^* = J_{AB} + J_B^*$, which is the overall cost, assuming optimal decision making. Thus, an optimal policy has been found which corresponds to the sequence of optimal decisions that has been determined.

Of course, one can wonder why in this case DP is a useful method. After all, the optimal solution to the problem shown in Figure 4.2 can be determined by simply comparing the paths A-B-C-E, A-B-D-E and A-B-E. However, in larger problems such an exhaustive search (trying all allowable paths) may be extremely computation-intensive. This becomes evident with regard to Figure 4.3, which shows the exact same staged decision process but doubled and put in series. The computational load using DP would simply double, compared to the example above. However, the number of paths to consider, which is now 9 instead of 3, would triple.[16]

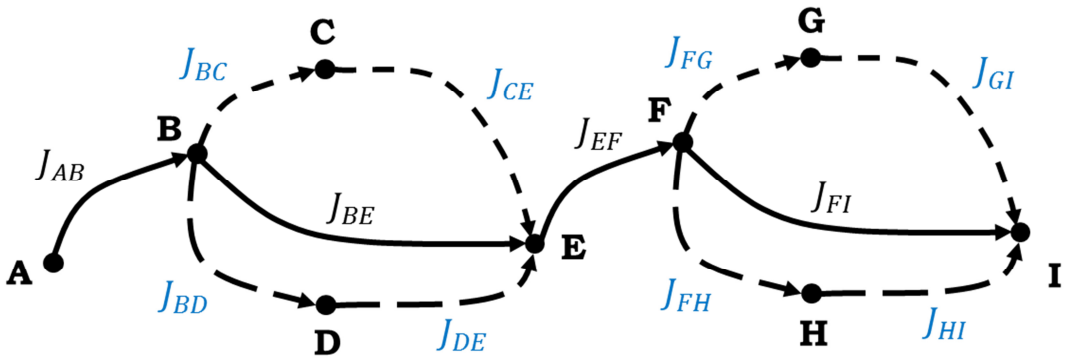

Figure 4.3: Another multistage decision problem.
Adapted from Kirk D. (2004): pp. 54.

Another important aspect is the so called Principle of Optimality. In 1954 Richard Bellman wrote: *"An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."*[17] Once an optimal policy for the problem in Figure 4.3 has been computed, choosing another starting point than A will

[16] Cf. Kirk D. (2004): pp. 56 - 58.

[17] Bellman R. (1954): pp. 4.

not result in any change of optimal decision making. And neither does violating the optimal policy because henceforth, previously computed optimal choices are still valid.[18]

General problem formulation:

Now that the fundamental idea of DP has been explained a basic model that has two key features is considered: (1) a discrete time dynamic system and (2) an additive cost function. The system is given by

$$x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0,1,\dots,N-1, \tag{4.2}$$

where

k is the discrete time index,

x_k is the state of the system at the k-th time step. Note that $x_k \in X_k$ is discrete both in time and value.

u_k indicates the (constrained) discrete-value control input or decision made at time k. The set of all feasible values can depend on the system state such that $u_k \in U_k(x_k)$.

w_k is a (random) parameter that represents a disturbance or noise,

N is the number of times the control input is applied and a decision is made, respectively and

f_k is a function that describes the system.

It is important to note that for the application of DP at least the probability distribution of the disturbance must be known a priori (stochastic dynamic programming). Furthermore, w_k may explicitly depend on both x_k and u_k but must be independent of prior values $w_{k-1},\dots,w_0$ (Markov process). In the next chapters (which deal with the implementation of velocity profiles), it will be assumed that every value of w_k is exactly known in advance (deterministic dynamic programming). However, for

[18] Cf. Kirk D. (2004): pp. 54.

the sake of generality w_k will be considered as a random variable in the following general problem formulation.

If $g_N(x_N)$ is some terminal cost incurring at the end of the process, the cost is

$$g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k). \tag{4.3}$$

Since the disturbance may be a random parameter, the cost is generally a random variable and can therefore not be meaningfully optimized. Thus, the problem can be formulated as an optimization of the expected cost[19]

$$E\left\{g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k)\right\}. \tag{4.4}$$

Let $\pi = (\mu_0, \dots, \mu_{N-1})$ be any admissible policy (control sequence) such that μ_k maps every state x_k into a control input $u_k = \mu_k(x_k)$. With $\pi \in \Pi$, Π denotes the set of all admissible policies. The cost of using π on problem (4.2) with the initial condition x_0 is defined by

$$J_\pi(x_0) = E\left\{g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k)\right\}. \tag{4.5}$$

The optimal policy π_{opt} then minimizes J_π

$$J_{opt}(x_0) = \min_{\pi \in \Pi} J_\pi(x_0). \tag{4.6}$$

Now the following algorithm (which proceeds backwards from $N-1$ to 0) can be executed to determine the optimal cost at every time k and the optimal policy π_{opt}:

$$J_k(x_k) = \min_{u_k \in U_k(x_k)} E\{g_k(x_k, u_k, w_k) + J_{k+1}(x_{k+1})\}, \tag{4.7}$$

[19] Cf. Bertsekas D. (2005): pp. 3-4.

where $J_N(x_N) = g_N(x_N)$.[20]

Note that the terminal cost $g_N(x_N)$ can be used to compute π_{opt} in a way such that a desired final state is $x_{N,desired}$ is approached. Let $x_{tol} > 0$ specify a tolerance range for the final state x_N around $x_{N,desired}$. By choosing

$$g_N(x_N) = \begin{cases} 0 & for\ |x_N - x_{N,desired}| < x_{tol} \\ \alpha & for\ |x_N - x_{N,desired}| \geq x_{tol}, \end{cases} \tag{4.8}$$

where $\alpha \in \mathbb{R}$ must be a sufficiently high number, the final state x_N will be close to $x_{N,desired}$.

The formulation of the basic problem and the DP algorithm is hereby concluded. This chapter will therefore proceed with some implementation issues, which will be useful for the computation of energy-optimal velocity profiles in the next subchapter.

Grid selection:

Dynamic programming can only be applied on systems with limited discrete control inputs (a limited number of possible decisions) and limited discrete system states. In the introductory example (Figure 4.2) this requirement was fulfilled automatically. However, in many dynamic systems both the number of feasible control inputs, and the number of states is infinite. This requires a discretization as a first step. The number of grid elements represents a tradeoff between high-accuracy and computing time.[21]

[20] Cf. Guzzella L., Sciarretta A. (2007): pp. 313-315.

Cf. Bertsekas D. (2005): pp. 12-13.

[21] Cf. Guzzella L., Sciarretta A. (2007): pp. 316.

Interpolation between two grid points:

The discretization of the state space causes the following issue: if at $x_k \in X_k$ the next state is calculated using $x_{k+1} = f_k(x_k, u_k, w_k)$ that state might not exactly match one of the possible states in X_{k+1}. This causes a problem, because the so called cost-to-go $J_{k+1}(x_{k+1})$ in equation (4.7) is only defined for any $x_{k+1} \in X_{k+1}$. Figure 4.4 shows this problem, where for simplicity $X_k = X_{k+1}\ \forall k$ and $w_k = 0\ \forall k$.

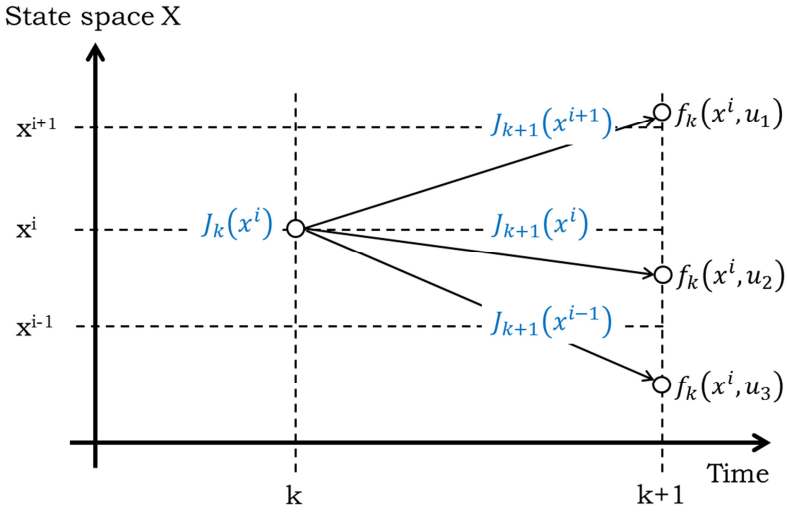

Figure 4.4: Issues arising due to discrete state space.

Adapted from: Guzzella L., Sciarretta A. (2007): pp. 317.

The number of valid control inputs is limited to three (u_1, u_2, u_3). At time k, the state x^i is considered. All inputs have to be considered to calculate $J_k(x_k)$, which is according to equations (4.2) and (4.7) determined by

$$J_k(x_k) = \min \begin{Bmatrix} g_k(x^i, u_1) + J_{k+1}\left(f_k(x^i, u_1)\right) \\ g_k(x^i, u_2) + J_{k+1}\left(f_k(x^i, u_2)\right) \\ g_k(x^i, u_3) + J_{k+1}\left(f_k(x^i, u_3)\right) \end{Bmatrix}. \tag{4.9}$$

However, as mentioned above, J_{k+1} is not defined for those values because they do not match with X_k. For this reason some approximation must be found. An easy solution is to simply set J_{k+1} to the closest defined value. This is called the nearest neighbor method and its advantage is a low computational load.

A second approach (which is also used in this thesis for the computation of optimal velocity profiles) is to use a linearly interpolated value of the cost-to-go J_{k+1}. Other interpolation methods can be used as well, but linear interpolation usually results in a reasonable tradeoff between high accuracy and the extra computational load. In this case the approximation of equation (4.9) is[22]

$$J_k(x_k) = \\ = \min \begin{Bmatrix} g_k(x^i, u_1) + J_{k+1}(x^{i+1}) + \left(f_k(x^i, u_1) - x^{i+1}\right)\dfrac{J_{k+1}(x^{i+2}) - J_{k+1}(x^{i+1})}{x^{i+2} - x^{i+1}} \\ g_k(x^i, u_2) + J_{k+1}(x^{i-1}) + \left(f_k(x^i, u_2) - x^{i-1}\right)\dfrac{J_{k+1}(x^{i}) - J_{k+1}(x^{i-1})}{x^{i} - x^{i-1}} \\ g_k(x^i, u_3) + J_{k+1}(x^{i-2}) + \left(f_k(x^i, u_3) - x^{i-2}\right)\dfrac{J_{k+1}(x^{i-1}) - J_{k+1}(x^{i-2})}{x^{i-1} - x^{i-2}} \end{Bmatrix}. \tag{4.10}$$

Infeasible states or inputs:

Infeasible states or inputs can be handled by assigning an infinite cost. A disadvantage of this method is that if interpolation is used, the interpolated value is infinity as well. If not treated correctly, this can cause additional cost-to-go values to become infinity. In this way infinite values can spread far into feasible state space. To avoid this, a very large real constant can be used, which has to be greater than any cost-to-go.[23]

4.1.1 Implementation

This section shows the implementation of a dynamic programming algorithm in MATLAB/Simulink to compute energy-optimal velocity profiles.

---

[22] Cf. Guzzella L., Sciarretta A. (2007): pp. 316 - 317.

[23] Cf. Guzzella L., Sciarretta A. (2007): pp. 318.

All simulations use only parameters taken from the prototypes that are described in chapter 2.

The way a dynamic programming algorithm is implemented has a huge impact on the computational load. An easy option to implement DP would be to use three for loops: one for every time step, one for every grid point and one for every admissible control input. However, using a triple-for loop in MATLAB is quite inefficient in terms of computing time. For this reason a vector-based approach is taken, which reduces the number of for loops to just one. This requires some extra implementation effort but the speedup factor is so significant that it will pay off.[24]

An interesting aspect is that those vectors never change during the execution of the dynamic programming algorithm. Thus, they can be calculated a-priori (details on this will be given below). Figure 4.5 shows the flow chart to create optimal velocity profiles. Note that the blocks on the left serve only the purpose to preprocess data and to put them in vectors to efficiently execute the dynamic programming algorithm. The last step is the forward simulation to create velocity profiles according to the optimal policy DP computes. All these steps in Figure 4.5 will now be explained below.

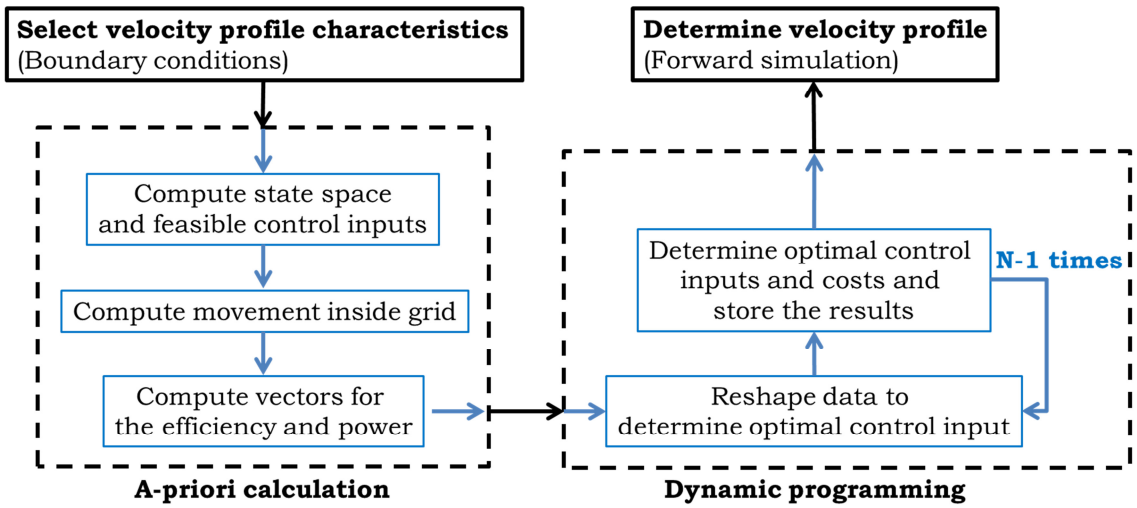

Figure 4.5: Flow chart to compute velocity profiles

[24] Cf. Guzzella L., Sciarretta A. (2007): pp. 318 - 319.

Select velocity profile characteristics (problem description):

As mentioned in chapter 4, velocity profiles must respect boundary conditions that can either be given at an instance in time ($v_0 = v(t = t_0)$, $v_N = v(t = t_N)$) or at a certain distance ($v_0 = v(d = d_0)$, $v_N = v(d = d_N)$). Accordingly, it is useful to view the vehicle velocity as a function of time or the traveled distance. However, in the introductory chapter of dynamic programming the index k only indicated discrete time steps. Therefore, the first step is to extend this notation. From now on it will be evident from the context, whether k indicates discrete time or discrete distance. The algorithm automatically generates either a distance or time grid, depending on the boundary conditions.

The discrete-time dynamic system is given by $v_{k+1} = f(v_k, M_k)$. Let Δt_k be the time between the steps k and +1 . Then the system equation is defined as

$$v_{k+1} = v_k + a_k \Delta t_k, \tag{4.11}$$

where a_k is the vehicle acceleration, which according to equations (2.2) and (2.7) is given by

$$a_k = \frac{r_{wheel}}{I_{veh}} \left(M_k i_{tr} \eta_{tr} \eta_{diff} - M_{res,k} \right). \tag{4.12}$$

However, Δt_k depends on whether (i) a time grid or (ii) a distance grid is used.

i) In case a time grid is used Δt_k is explicitly specified as the step time of the DP algorithm, which is the same for all k. Hence, $\Delta t_k = \Delta t_{k+1}$

ii) If a distance grid is used a fixed distance step Δd is defined within the DP algorithm. Therefore the time to travel from one point of the distance grid to the next has to be determined. If constant acceleration is asserted, the distance that an object moves within the time Δt_k is given by

$$\Delta d = v_k \Delta t + \frac{1}{2} a_k \Delta t_k^2. \tag{4.13}$$

Solving this equation for Δt gives

$$\Delta t_k = \begin{cases} -\frac{v_k}{a_k} - \sqrt{\frac{v_k^2}{a_k^2} + 2\frac{\Delta d}{a_k}} & for\ a_k < 0 \\ -\frac{v_k}{a_k} + \sqrt{\frac{v_k^2}{a_k^2} + 2\frac{\Delta d}{a_k}} & for\ a_k > 0 \\ \frac{\Delta d}{v_k} & for\ a_k = 0. \end{cases} \quad (4.14)$$

It is possible that at low velocities and low torques the discriminant becomes negative. This case is handled by setting Δt_k to some arbitrary real value. This is a valid method because the control input that causes v_{k+1} to be negative will have a high cost function and therefore not be selected as an optimal control input (details will follow below). Also, infinite numbers for Δt_k result in high cost functions because $0 \notin V_k$.

Compute state space and feasible control inputs:

The vehicle velocity v_k is a continuous state variable (it is only discrete in time). Therefore it is necessary to discretize $v_k \in V_k$ in value. An equidistant discretization is chosen, where the gap between two grid points is $\Delta v \approx 0.3\, km/h$. Furthermore, the velocity grid does not change over k. Hence, $V_k = V_{k+1}\ \forall k$.

The constraint for the vehicle speed in equation (4.1) $v_{min} \leq v(\xi) \leq v_{max}$ can easily be implemented by limiting V_k. Assuming that $v_{min} + n\Delta v = v_{max}$, where $n \in \mathbb{Z}^+$, the state space V_k can be represented in MATLAB by a vector $\tilde{\boldsymbol{\gamma}}$:

$$\tilde{\boldsymbol{\gamma}}^T = [v_{min} \quad v_{min} + \Delta v \quad v_{min} + 2\Delta v \quad \dots \quad v_{min} + (n-1)\Delta v \quad v_{max}]. \quad (4.15)$$

The next step is to define feasible values for the control input which is the torque of both electric machines. On the one hand these values are constrained by the minimum and maximum available torque depending on the speed ($M_{min,EM}(v)$, $M_{max,EM}(v)$). On the other hand the torque may be limited due to comfort reasons ($M_{max,comf}$) or by the ESPhev charac-

teristics. The latter is not yet expressed as a torque constraint. Fortunately equations (2.2) and (2.7) state a clear relation between the vehicle acceleration and the overall torque. Thus, the ESPhev limit for pure regenerative breaking $\dot{v}_{min,ESP} = -1.25\, m/s^2$ can also be expressed by a velocity-dependent torque constraint:

$$M_{min,ESP}(v) = \left(\dot{v}_{min,ESP} \frac{I_{veh}}{r_{wheel}} + M_{res}(v) \right) \frac{1}{i_{tr} \eta_{tr} \eta_{diff}}. \tag{4.16}$$

Equation (4.16) must be evaluated for every element in $\tilde{\boldsymbol{\gamma}}$.

The range of feasible control inputs for the i-th velocity in $\tilde{\boldsymbol{\gamma}}$ is therefore defined by the bounds

$$\begin{aligned} M^i_{min} &= \max\{M^i_{min,EM}, M^i_{min,ESP}\}, \\ M^i_{max} &= \min\{M^i_{max,EM}, M^i_{max,comf}\}, \end{aligned} \tag{4.17}$$

where $i = 1, \dots, n+1$.

Now, a set of vectors is defined by

$$\begin{aligned} \tilde{\boldsymbol{\vartheta}}^{i-T} &= \left[M^i_{min} \quad M^i_{min} \frac{\alpha_1 - 1}{\alpha_1} \quad M^i_{min} \frac{\alpha_1 - 2}{\alpha_1} \quad \dots \quad M^i_{min} \frac{1}{\alpha_1} \right], \\ \tilde{\boldsymbol{\vartheta}}^{i+T} &= \left[M^i_{max} \frac{1}{\alpha_2} \quad \dots \quad M^i_{max} \frac{\alpha_2 - 1}{\alpha_2} \quad M^i_{max} \right], \\ \tilde{\boldsymbol{\vartheta}}^{iT} &= [\tilde{\boldsymbol{\vartheta}}^{i-T} \quad 0 \quad \tilde{\boldsymbol{\vartheta}}^{i+T}], \end{aligned} \tag{4.18}$$

where $\alpha_1, \alpha_2 \in \mathbb{Z}^+$ are parameters to define the number of feasible control inputs at each discrete velocity. Herby α_1 defines the number of discretized negative torque values and the positive torque values are defined by α_2 (see equation (4.18)). The total length of $\tilde{\boldsymbol{\vartheta}}^i$ is given by $\alpha_1 + \alpha_2 + 1$. Of course, only one parameter would be sufficient to compute $\tilde{\boldsymbol{\vartheta}}^i$ but usually $|M^i_{min}| \ll M^i_{max}$. Therefore, by using α_1 and α_2 it is possible to achieve similar grid accuracy in positive and negative torque regions.

Finally, *all* feasible control inputs are summarized in the vector $\boldsymbol{\vartheta}$:

$$\boldsymbol{\vartheta}^T = [\tilde{\boldsymbol{\vartheta}}^{1T} \quad \tilde{\boldsymbol{\vartheta}}^{2T} \quad \dots \quad \tilde{\boldsymbol{\vartheta}}^{(n+1)T}]. \tag{4.19}$$

Note that the length and therefore the total number of control inputs in every step k is $(n+1)(\alpha_1+\alpha_2+1)$.

Calculate movements inside the grid:

Due to the fact that all feasible values for the velocity as well as all feasible control values are known a priori, *all* possible changes of the velocity between k and $k+1$ for every k can be computed in advance. Let $\boldsymbol{\gamma}$ be a vector in which all elements of $\tilde{\boldsymbol{\gamma}}$ are repeated $\alpha_1+\alpha_2+1$ times.

$$\boldsymbol{\gamma} = \begin{bmatrix} v_{min} \\ \vdots \\ v_{min} \\ v_{min}+\Delta v \\ \vdots \\ v_{min}+\Delta v \\ \vdots \\ v_{max} \\ \vdots \\ v_{max} \end{bmatrix} \begin{matrix} \left.\begin{matrix}\\\\\end{matrix}\right\} (\alpha_1+\alpha_2+1) \\ \left.\begin{matrix}\\\\\end{matrix}\right\} (\alpha_1+\alpha_2+1) \\ \}(n-2)(\alpha_1+\alpha_2+1) \\ \left.\begin{matrix}\\\\\end{matrix}\right\} (\alpha_1+\alpha_2+1) \end{matrix} \tag{4.20}$$

Now both $\boldsymbol{\gamma}$ and $\boldsymbol{\vartheta}$ have the exact same number of elements.

Furthermore let $\Delta\boldsymbol{t}$ be a vector whose j-th element corresponds to the time it takes to move from one grid point if the vehicle speed is the j-th element of $\boldsymbol{\gamma}$ and the applied torque is the j-th element of $\boldsymbol{\vartheta}$ (if the dynamic programming algorithm uses a time grid, all elements of $\Delta\boldsymbol{t}$ are simply the sampling time and if a distance grid is used, $\Delta\boldsymbol{t}$ can be computed according to equation (4.14)).

If the control inputs specified in $\boldsymbol{\vartheta}$ are applied to the system with the states specified in $\boldsymbol{\gamma}$, the states change according to equation (4.11). *All* changed states can now be computed in only one step:

$$\boldsymbol{\gamma}_{next} = \boldsymbol{\gamma} + \frac{r_{wheel}}{I_{veh}}\left(i_{tr}\eta_{tr}\eta_{diff}\boldsymbol{\vartheta} - M_{res}(\boldsymbol{\gamma})\right) \circ \Delta\boldsymbol{t}, \tag{4.21}$$

where $\circ$ denotes an element-wise vector multiplication.

Keep in mind that the velocity has been discretized and that therefore – as explained in chapter 4.1 – the elements in $\boldsymbol{\gamma}_{next}$ might not exactly

match one of the predetermined values in $\tilde{\boldsymbol{\gamma}}$. That is the reason why interpolation (or at least a nearest neighbor approximation) of the cost-to-go functions is mandatory. In order to expedite the interpolation it is useful to precalculate $\boldsymbol{\gamma}_{next}^{-}$ and $\boldsymbol{\gamma}_{next}^{+}$. In $\boldsymbol{\gamma}_{next}^{-}$ each element of $\boldsymbol{\gamma}_{next}$ is decreased to the nearest possible state in $\tilde{\boldsymbol{\gamma}}$. In case this is not possible because no lower feasible state exists, the nearest (upper) value is chosen. Accordingly, $\boldsymbol{\gamma}_{next}^{+}$ contains rounded up elements of $\boldsymbol{\gamma}_{next}$.

Finally, two auxiliary vectors are introduced – again to speed up the interpolation below. Let the elements of $\boldsymbol{m}^{-}$ and $\boldsymbol{m}^{+}$ represent the number of grid points Δv between $\boldsymbol{\gamma}$ and $\boldsymbol{\gamma}_{next}^{-}$ and $\boldsymbol{\gamma}_{next}^{+}$, respectively:

$$\boldsymbol{m}^{\pm} = \frac{1}{\Delta v}\left(\boldsymbol{\gamma}_{next}^{\pm} - \boldsymbol{\gamma}\right). \tag{4.22}$$

Note that as a result of the definitions of the variables above the elements of $\boldsymbol{m}^{\pm}$ are guaranteed to be integers and that (again by definition) the elements of $(\boldsymbol{m}^{+} - \boldsymbol{m}^{-})$ are either 0 or 1.

What is still missing is some kind of dealing with infeasible states. As a reminder, infeasible states are states that are above v_{max} or below v_{min}. As mentioned in chapter 4.1 a viable solution is to assign a sufficiently high cost to those states. For this purpose a vector $\boldsymbol{J}_{inf}$ is created, whose j-th element is defined by

$$J_{inf}^{j} = \begin{cases} \alpha_{inf} & for\ \gamma_{next}^{j} < v_{min}\ or\ \gamma_{next}^{j} > v_{max} \\ 0 & else, \end{cases} \tag{4.23}$$

where α_{inf} is a sufficiently high real number, which by default is arbitrarily set to 10^6.

Compute vectors for the efficiency and power:

The supply voltage as well as the rotor and stator temperatures are considered constant and equal for both electric machines. Hence, the efficiency only depends on the velocity and torque. Let $\tilde{\eta}_{2EM,subopt}(v, M)$ be the combined electric machine efficiency according to the torque split factor $u_{f,subopt}$. To obtain the electrical power, the mechanical power

must be multiplied by $\tilde{\eta}_{2EM,subopt}$ for $M < 0$ and divided by $\tilde{\eta}_{2EM,subopt}$ for $M \geq 0$, respectively. However, this case distinction would lengthen the notation below. Therefore, it is useful to define $\eta_{2EM,subopt}$ as

$$\eta_{2EM,subopt}(v, M) = \begin{cases} \tilde{\eta}_{2EM,subopt}(v, M) & for\ M < 0 \\ \dfrac{1}{\tilde{\eta}_{2EM,subopt}(v, M)} & for\ M \geq 0. \end{cases} \tag{4.24}$$

Since $\boldsymbol{\gamma}$ and $\boldsymbol{\vartheta}$ hold all valid velocity-torque combinations in a specified order, again, a new vector $\boldsymbol{\eta}$ can be defined whose j-th element corresponds to $\eta_{2EM,subopt}$ at the velocity value in the j-th element of $\boldsymbol{\gamma}$ and the torque value in the j-th element of $\boldsymbol{\vartheta}$. Hence,

$$\boldsymbol{\eta} = \eta_{2EM,subopt}(\boldsymbol{\gamma}, \boldsymbol{\vartheta}). \tag{4.25}$$

The electrical power of both EM is

$$\boldsymbol{P}_{2EM} = \frac{1}{r_{wheel}} \boldsymbol{\gamma} \circ \boldsymbol{\vartheta} \circ \eta_{2EM,subopt}(\boldsymbol{\gamma}, \boldsymbol{\vartheta}). \tag{4.26}$$

With the electric current of both EM being

$$\boldsymbol{I} = \boldsymbol{P}_{2EM} \frac{1}{V_{DC}}, \tag{4.27}$$

the power loss in the battery can be calculated by

$$\boldsymbol{P}_{Batt} = \boldsymbol{I} \circ \boldsymbol{I} R, \tag{4.28}$$

where R is the battery resistance. Finally, the total energy consumption is given by

$$\boldsymbol{E} = (\boldsymbol{P}_{2EM} + \boldsymbol{P}_{Batt}) \circ \Delta \boldsymbol{t}. \tag{4.29}$$

Note that so far the dynamic programming algorithm has not yet started but all a-priori calculations have now been concluded. A short summary of the essential variables is given:

- $\boldsymbol{\gamma}$ contains the velocity values of the speed grid whereat every element is repeated $n + 1$ times.
- The elements of $\boldsymbol{\vartheta}$ correspond to (piecewise) linearly distributed torque values between the minimum and maximum torque at any speed in $\boldsymbol{\gamma}$.

- At the k-th time or distance step, let a constant torque value according to the j-th element of $\boldsymbol{\vartheta}$ be applied at a velocity that corresponds to j-th element in $\boldsymbol{\gamma}$. In this case the j-th element in $\boldsymbol{\gamma}_{next}$ corresponds to the velocity at the subsequent (k+1)-th step.
- Due to the fact that in general the elements of $\boldsymbol{\gamma}_{next}$ do not match the velocity grid, $\boldsymbol{\gamma}_{next}^{-}$ and $\boldsymbol{\gamma}_{next}^{+}$ are introduced and will be used in the cost-to-go interpolation below. Furthermore $\boldsymbol{m}^{\pm}$ represent the number of grid points Δv between $\boldsymbol{\gamma}$ and $\boldsymbol{\gamma}_{next}^{\pm}$. In addition, a sufficiently large cost is assigned to infeasible states through $\boldsymbol{J}_{inf}$.
- The elements of $\boldsymbol{E}$ contain the total electrical energy that is consumed between two steps k and k+1.

Interpolate cost-to-go and compute total cost:

Dynamic programming algorithms proceed backwards from $k = N$ to $k = 0$. First, at the N-th step the boundary condition for the final vehicle speed v_N must be taken into account. Let $\gamma_{next,j}$ be the j-th element of $\boldsymbol{\gamma}_{next}$ and $g_{N,j}$ be the j-th element of the terminal cost $\boldsymbol{g}_N = g_N(\boldsymbol{\gamma}_{next})$. Based on equation (4.8) the terminal cost can be specified by

$$g_{N,j}(\gamma_{next,j}) = \begin{cases} 0 & for\ |\gamma_{next,j} - v_N| < v_{tol} \\ \alpha & for\ |\gamma_{next,j} - v_N| \geq v_{tol}, \end{cases} \tag{4.30}$$

where the default setting for the tolerance speed is $v_{tol} = 0.1 m/s$ and α is arbitrarily set to $5*10^5$. Note that in the first step the cost-to-go function is equal to the terminal cost function and that $\boldsymbol{g}_N$ is again a column vector with $(n+1)(\alpha_1+\alpha_2+1)$ elements.

For any other step k $(0 \leq k \leq N-1)$ the cost-to-go function is determined by interpolation. When implementing a dynamic programming algorithm, one should keep in mind that MATLAB contains several

built-in interpolation functions. However, their calculating time is quite high compared to a properly made custom interpolation.[25]

As a starting point, imagine the vehicle is moving at step k at a velocity v_k^j (j indicates any index in the extended velocity grid $\boldsymbol{\gamma}$). Due to the definition of $\boldsymbol{m}^{\pm}$, the velocity at k+1 will be within the interval $v_k^j + m^{+,j}\Delta v \leq v_{k+1}^j \leq v_k^j + m^{-,j}\Delta v$. Thus, it is evident that in order to calculate J_k^j an interpolation between $J_{k+1}^{j+m^{+,j}}$ and $J_{k+1}^{j+m^{-,j}}$ is necessary. The latter two are merged in vectors and represent the j-th elements in $\boldsymbol{J}_{k+1}^{+}$ and $\boldsymbol{J}_{k+1}^{-}$, respectively. Both simply represent rearranged versions of $\boldsymbol{J}_{k+1}$. Since $\boldsymbol{m}^{\pm}$ has already been calculated in advance, this rearrangement can be computed very efficiently in MATLAB.

The computation of the total cost can now be implemented with just one final equation:

$$\boldsymbol{J}_k = \boldsymbol{E} + \boldsymbol{J}_{inf} + \boldsymbol{J}_{k+1}^{-} + (\boldsymbol{\gamma}_{next} - \boldsymbol{\gamma}_{next}^{-}) \circ (\boldsymbol{J}_{k+1}^{+} - \boldsymbol{J}_{k+1}^{-}) \oslash (\boldsymbol{\gamma}_{next}^{+} - \boldsymbol{\gamma}_{next}^{-}). \quad (4.31)$$

Note that $\oslash$ indicates a Hadamard (element wise) vector division.

Determine optimal control inputs and costs and store the results:

The total costs for every velocity-torque combination are now stored in $\boldsymbol{J}_k$ and somehow the optimal control input for every velocity must be determined. In MATLAB this can be done quite simply. By using the command *reshape* $\boldsymbol{J}_k$ is converted to a $(\alpha_1+\alpha_2+1) \times (n+1)$ matrix. The columns represent the costs for the control inputs and the rows represent the velocity grid. The command *min* then returns the indices of every optimal control input as well as the corresponding optimal cost-to-go.

The results are the vectors $\boldsymbol{J}_{opt,k}$ and $\boldsymbol{M}_{opt,k}$. They represent the columns of the matrices $\boldsymbol{J}_{opt}$ and $\boldsymbol{M}_{opt}$, respectively.

[25] Cf. Guzzella L., Sciarretta A. (2007): pp. 318.

Determine velocity profile (forward simulation):

The optimal velocity profiles are computed in Simulink. For clarity, a simplified version of the Simulink model is explained that only computes velocity profiles for distance grids (see Figure 4.6). There are two integrators whose outputs are the velocity and the traveled distance. The driving torque and the resistance forces are converted to acceleration and form the input of the first integrator.

The optimal control inputs are stored in the look-up table. Its inputs are the current position and velocity. In this block, the setting *Interpolation – Use End Values* is enabled to improve the simulation accuracy. In reality the interpolated optimal control input will only be computed with a specific sample time. To emphasize this sampling, a Zero-Order-Hold block has been included to the model.

The low-pass filter fulfills two purposes:

i) The low-pass filter limits the rise time because the optimal control input cannot be applied arbitrarily fast.

ii) The amplitude of an energy-optimal control input can change very abruptly which might feel unpleasant to the driver. Therefore, the low-pass is also important to increase the traveling comfort.

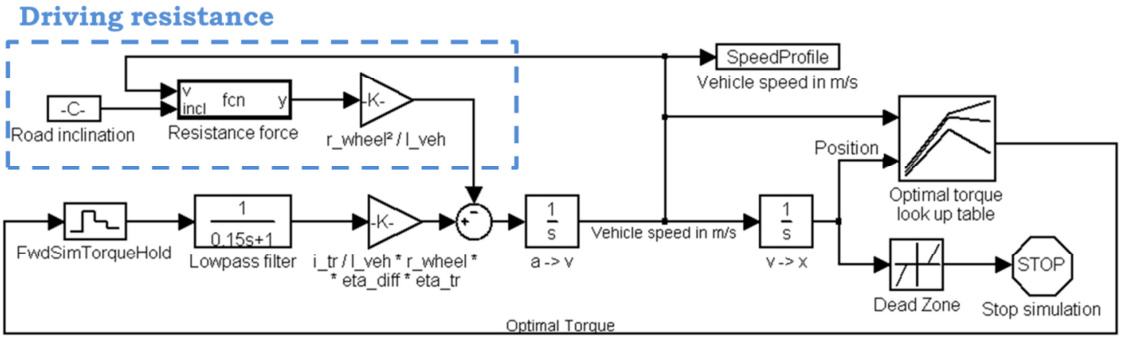

Figure 4.6: Forward simulation Simulink model (simplified, for velocity trajectories that are functions of the traveled distance).

Note that in case the dynamic programming algorithm uses a time grid, the second input of the look-up table must be the simulation time instead of the position.

4.1.2 Results and discussion

In this section two particular use cases for energy-optimum velocity profiles are considered. Example 1 demonstrates a deceleration maneuver that is computed by the DP algorithm using a distance grid. In example 2 an acceleration maneuver is performed that takes a specific period of time.

Example 1:

Consider a scenario in which a vehicle is traveling on a rural road at 100 km/h and is approaching a village where a speed limit of 50 km/h must be adhered. The vehicle speed is not to be reduced until 500m in front of the village. The boundary conditions are given by $v(\xi_0) = 100\,km/h$ and $v(\xi_N = 500m) = 50\,km/h$. The velocity grid is (arbitrarily) generated between $v_{min} = 16\,km/h$ and $v_{max} = 103\,km/h$.

Figure 4.7 shows the resulting velocity profile. Even though it is computed using a distance grid it is chosen – for better visibility – to show the velocity as a function of the time. The resulting maneuver time is about 28 seconds and cannot directly be influenced by constraints. The velocity trajectory can roughly be divided into several phases. These phases were *empirically* discovered by evaluating several velocity profiles where $v(\xi_0) \geq v(\xi_N)$. Note however, that the algorithm can also compute optimal velocity profiles for $v(\xi_0) < v(\xi_N)$.

i) Regenerative breaking: preferably energy is recuperated at high velocities because of the greater driving resistance.
ii) Freewheeling: the driving torque is exactly zero for a significant period of time.
iii) Acceleration: due to lower driving resistance, traveling at low speed is more energy efficient than traveling at high velocity. If the maneuver time or distance is sufficiently large, the target velocity will be undercut and shortly before the end of the maneuver a relatively high torque is applied to meet the boundary condition $v(\xi_N)$. However, this behavior might be unwanted in

many cases. It can easily be prevented by choosing v_{min} accordingly.

The third phase is barely visible in this example, but another use case will be explained below where this phenomenon can be observed more clearly.

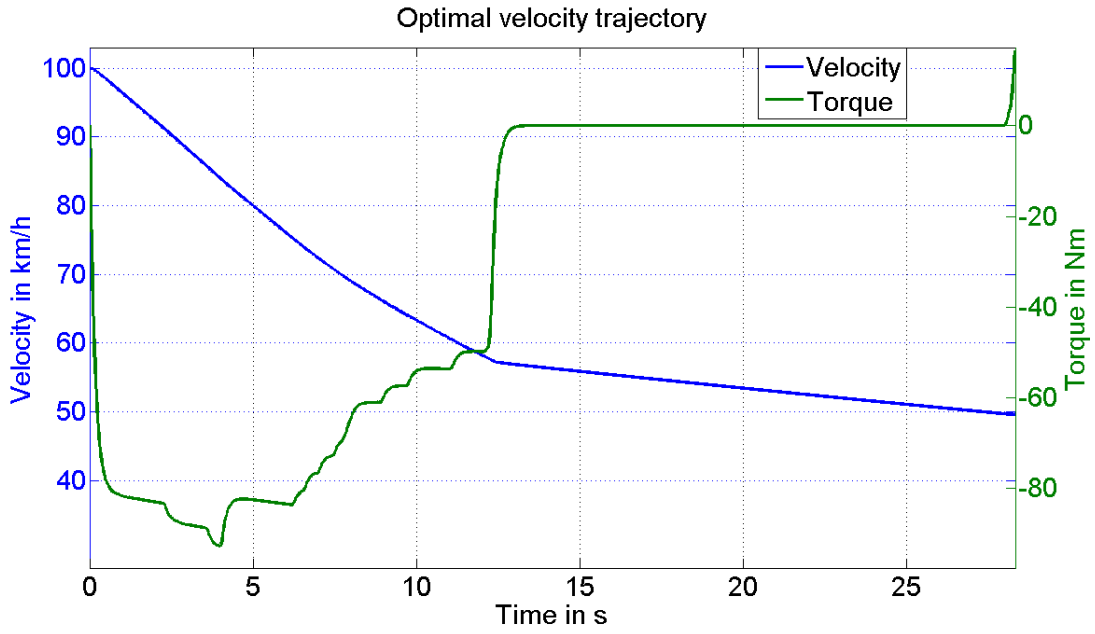

Figure 4.7: Energy-optimal velocity trajectory.

Based on this result lots of interesting theoretical aspects about dynamic programming that were discussed in chapter 4.1 can now be visualized. Consider Figure 4.8 where the optimal cost-to-go for every point inside the grid up to 499m is shown.

Apparently, the higher the initial velocity, the lower is the cost-to-go. At an initial speed of about 66km/h the SOC will not noticeably change. The cost-to-go is negative for any high initial velocity which means that the battery's state of charge will increase. At 100km/h the cost-to-go is about -4.05*10^5Ws. This value can be validated by exporting the velocity trajectory of Figure 4.7 to AVL CRUISE. The maneuver is carried out using a standard driver model to follow that trajectory. The resulting overall energy consumption is -3.76*10^5Ws. The difference can be explained by the level of detail of the vehicle model in AVL CRUISE, where some simplifications made in the DP algorithm are modeled more pre-

cisely (e.g. fluctuations of the battery voltage due to changing load conditions etc.).

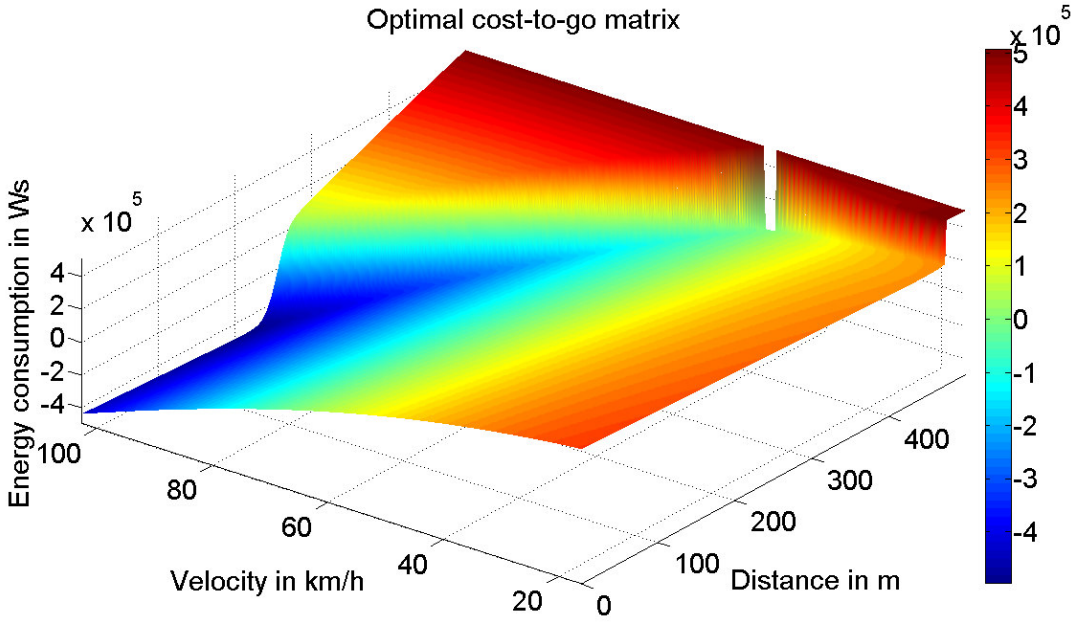

Figure 4.8: A cost-to-go matrix where $v(\xi_N)$=50 km/h.

Furthermore, the effect of the terminal cost stated in equation (4.30) can be observed. At the end of the maneuver there is a huge increase in the cost for any states that are not close to the boundary condition $v(\xi_N)$.

Figure 4.9 shows the optimum control inputs for any grid point and again the optimum velocity trajectory from Figure 4.7 but as a function of the traveled distance. The three phases mentioned above are distinguishable even though in this specific trajectory only the first two occur:

i) At higher velocities the optimum torque is clearly always negative because regenerative breaking optimal at high speed.

ii) At lower velocities the optimum torque is *exactly* zero in a large area of the grid.

iii) Due to freewheeling the vehicle might significantly undercut the target velocity $v(\xi_N)$. Therefore approaching ξ_N at low speed the optimum torque increases.

Note that in case the vehicle travels at a low velocity near the end of the maneuver it might not be possible to reach $v(\xi_N)$ because the vehicle

cannot accelerate arbitrarily fast. Nevertheless, in such a case the optimal control input is negative, which further increases the discrepancy to the boundary condition. But due to the definition of the terminal cost in equation (4.30) and the fact that regenerative breaking is applied the total cost is still minimized.

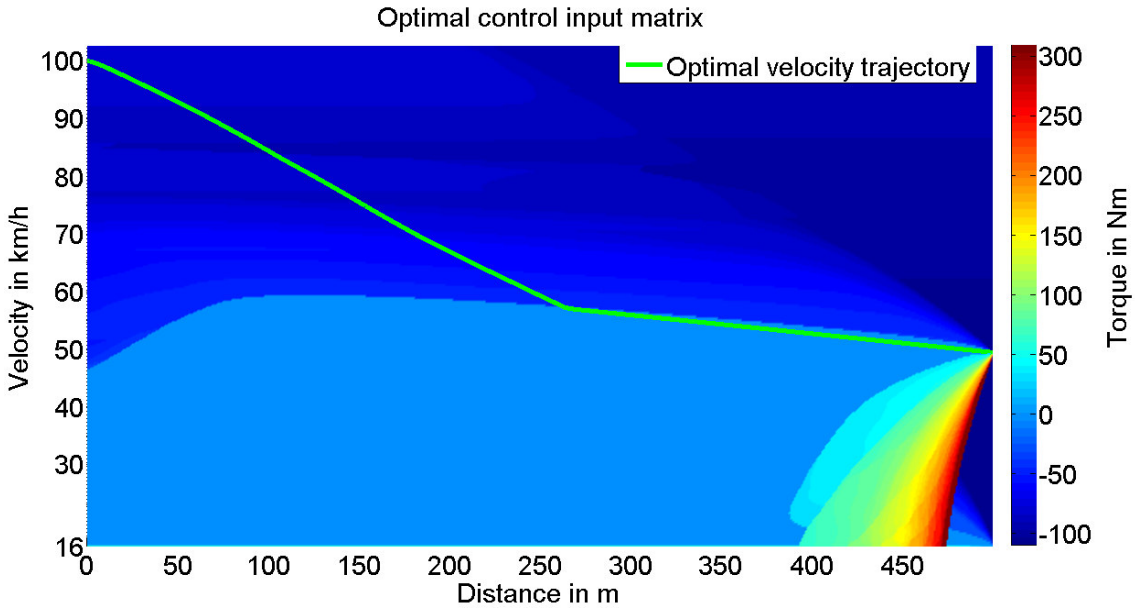

Figure 4.9: An optimal control input matrix where v(ξ_N)=50 km/h.

Since the cost to go matrix is invariant with regard to the initial point, an infinite number of energy-optimum trajectories can be computed that have the same boundary condition $v_N = v(\xi_N)$. This is shown in Figure 4.10 for a new cost-to-go matrix where $\xi_0 = 0m$ and $\xi_N = 750m$. As in the example above, the desired final velocity is 50km/h. However, the optimum control input matrix is computed for a 750m horizon. Trajectory 1 shows the maneuver being carried out in 750m. Now the previously mentioned third phase can be observed where the vehicle accelerates after undercutting the target velocity while freewheeling.

This new optimum control input matrix can also be used to reproduce the exact same result that was obtained above using the matrix that was computed for a 500m driving maneuver. Trajectory 2 represents copied data from Figure 4.9 whereas Trajectory 3 is created by using the new optimal control input matrix. It is obvious that the results are identical.

A fourth trajectory is shown where the initial speed is 30km/h. Even though this is already below the desired final velocity, according to the optimal policy the vehicle must freewheel at first but cannot undercut $v_{min} = 16\ km/h$.

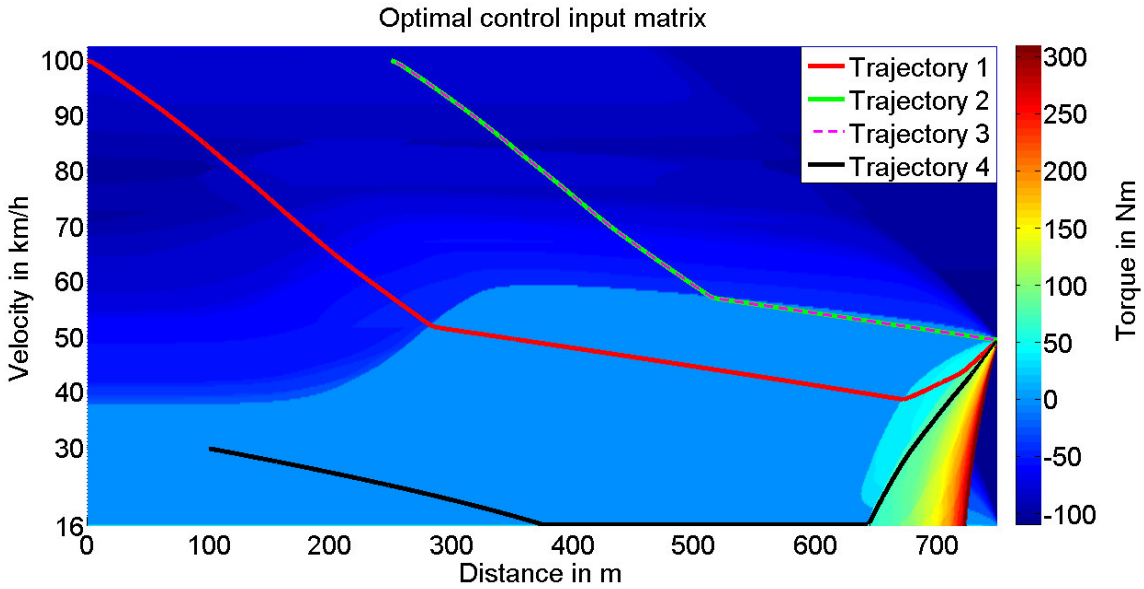

Figure 4.10: Relation between the optimal control input matrix and optimal trajectories.

From Figure 4.10 is evident that:

i) Any globally optimal trajectory is predetermined by the optimal control input matrix.

ii) For any valid starting point within the grid a globally optimum trajectory can be determined. Disturbances will be compensated automatically.

iii) According to Bellman's principle of optimality the optimal control input matrix for $\xi \geq 250$ in Figure 4.10 is identical to the one of Figure 4.9. Hence, Trajectory 2 and Trajectory 3 are identical.

Furthermore this method provides the opportunity to implement optimal velocity profiles with very little on-board computational effort. Several optimal control matrices for various final velocities can be computed in advance and stored in the vehicle. If $\xi_N - \xi_0$ is sufficiently high, any driving maneuver can be carried out where it is desired to reach a

specific velocity within a specified time or distance. Disturbances as for example bumpy roads or wind are compensated automatically. These properties make this approach quite robust.

Example 2:

In the following use case an energy-optimal acceleration maneuver is shown. Consider a motorist on a highway who drives through a construction site where the speed limit is 80km/h. As soon as the speed limit is lifted the motorist wants to accelerate to 120km/h. The acceleration maneuver shall take exactly 25 seconds. Thus, the boundary conditions are given by $v(\xi_0 = 0s) = 80km/h$, $v(\xi_N = 25s) = 120km/h$ and the velocity grid is generated between $v_{min} = 16km/h$ and $v_{max} = 130km/h$.

In Figure 4.11 the energy-optimum velocity trajectory is shown (blue curve, no constraints). Clearly, the vehicle is decelerating (freewheeling) for about 10 seconds down to 73.2km/h, before accelerating again up to 120km/h. Needless to say, decelerating at an increase of the speed limit might seem quite inconvenient to most drivers. It is therefore essential to impose a velocity constraint that prevents the vehicle speed from undercutting the initial speed.

An easy solution is to set $v_{min} = 80km/h$. Note that in this case the DP algorithm creates a velocity grid between 80km/h and 130km/h to compute an additional velocity trajectory that is also shown in Figure 4.11 (magenta curve, velocity constraint). It is energy-optimal to continue driving at 80km/h for about 11.2 seconds before accelerating. For $\xi > 11.2s$ the velocity trajectory then continuously approaches the one without the velocity constraint (blue curve). The torque peaks at about 120Nm.

Some drivers who highly appreciate driving comfort might still not be satisfied with the result. For this reason, in addition to the velocity constraint, the maximum torque is set to $M_{max,comf} = 90Nm$. Again, the result is shown in Figure 4.11 (red curve, velocity and torque constraint). The vehicle continues to drive at 80km/h for five seconds and then ac-

celerates smoothly to the desired velocity. In doing so, the overall electric machine torque never exceeds 90Nm.

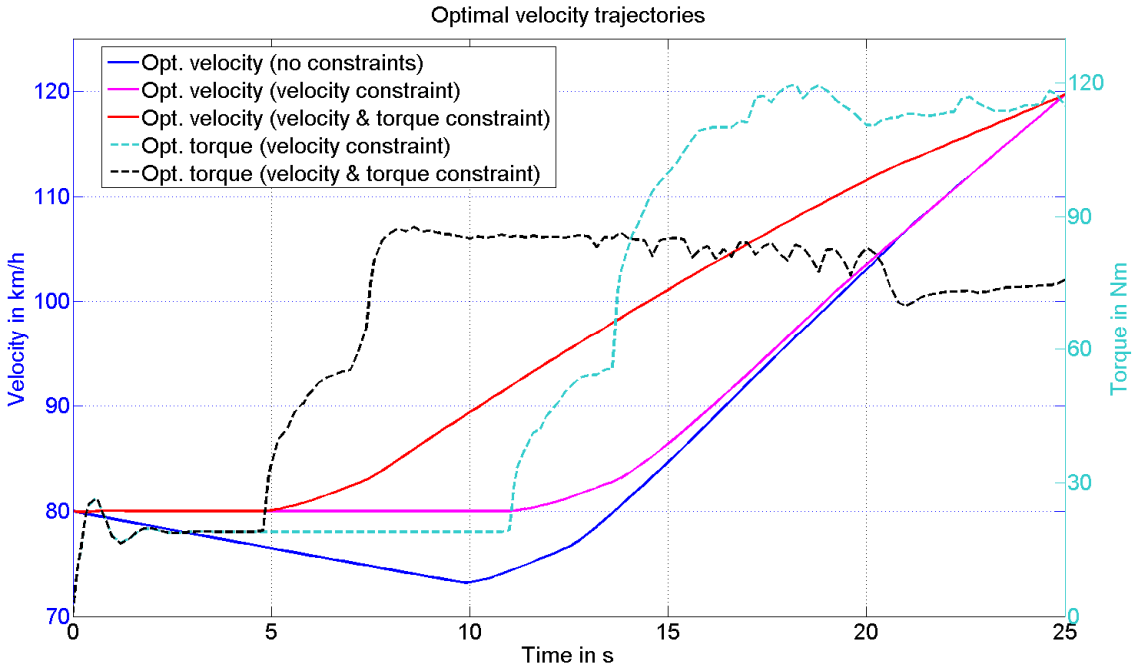

Figure 4.11: Energy-optimal velocity trajectories with and without constraints.

It is evident that by adding constraints to any minimization problem the optimum value of the cost function can only increase or remain unchanged. Thus, by limiting the minimum speed and the maximum torque, the energy consumption cannot decrease. On the other hand, the average velocity during the maneuver and therefore also the traveled distance rise as well. Table 4.1 shows the absolute and relative values of both the energy consumption and the traveled distance for all three energy-optimal velocity trajectories discussed above. All relative values refer to the unconstrained trajectory. It is interesting to point out that in both constrained cases the relative distance gain is higher than the relative increase of the energy consumption.

Energy consumption and traveled distance				
-	**Absolute energy**	**Relative energy**	**Absolute distance**	**Relative distance**
No constraints	1.15*10⁶Ws	100.0%	605.6m	100.0%
Velocity constraint	1.17*10⁶Ws	101.7%	622.7m	102.8%
Velocity & torque constraints	1.22*10⁶Ws	106.1%	668.5m	110.4%

Table 4.1: Comparison of energy-optimal velocity trajectories.

For a direct comparison of the energy consumption in relation to a specific traveled distance some additional information is needed. For this purpose, assume that after the driving maneuvers depicted in Figure 4.11 are completed the vehicle continues to travel at exactly 120km/h (33.$\dot{3}$m/s). In this state the power that is drained from the battery is approximately 2.94*10⁴W. Thus, the energy consumption per traveled meter is given by

$$P_{1m} = \frac{2.94 * 10^4 W}{33.\dot{3}\, m/s} = 8.82 * 10^2 W\, s/m. \tag{4.32}$$

Remember the highest traveled distance (668.5m) is achieved if both the velocity and the torque constraint are imposed and that in this case the energy consumption is 1.22*10⁶Ws. If no constraints apply, the vehicle travels $d_{add,1} = 668.5m - 605.6m = 62.9m$ less far. The energy that is necessary to travel the remaining 62.9m at 120km/h is

$$E_{add,1} = P_{1m} d_{add,1} = 8.82 * 10^2 W\, s/m * 62.9m \approx 5.55 * 10^4 Ws. \tag{4.33}$$

If only the constraint for the minimum velocity applies, the vehicle must travel $d_{add,2} = 668.5m - 622.7m = 45.8m$ at 120km/h. The consumed energy is

$$E_{add,2} = P_{1m} d_{add,2} = 8.82 * 10^2 W\, s/m * 45.8m \approx 4.04 * 10^4 Ws. \tag{4.34}$$

It is evident that

$$1.22 * 10^6 Ws > 1.17 * 10^6 Ws + E_{add,2} > 1.15 * 10^6 Ws + E_{add,1}. \tag{4.35}$$

The result shows that even in relation to a specific traveled distance the first (unconstrained) velocity trajectory still yields the lowest energy consumption. However, the relative energy saving for equal traveled dis-

tances is far less than the one in Table 4.1 where the relative energy consumption is compared at a specific maneuver time $\xi_N - \xi_0 = 25s$ and different distances. Also, keep in mind that in this example energy-optimal velocity profiles are compared among one another. The energy gain compared to a human driver must be investigated separately.

Note also that example 2 clearly demonstrates that there is a distinct tendency to travel at low speed as long as possible due to the lower driving resistance. This generally is true for not just this particular example but for all energy-optimal velocity profiles. Therefore, in any case where the initial velocity is lower than the desired velocity ($v(\xi_0) < v(\xi_N)$) it might be energy-optimal to first reduce the speed before accelerating to $v(\xi_N)$.

4.2 Velocity trajectory optimization using B-splines

The aim of this section is again to develop energy-optimum speed profiles but by applying a parameter optimization method. Any speed profile in this chapter is represented by a number of weighted base functions called B-Splines (see chapter 4.2.1 for details). The weighting factors of these B-splines represent the optimization parameters. Several constraints and boundary conditions apply which are similar to the ones mentioned in dynamic programming in chapter 4.1. Also, an exact length of the maneuver must be specified. This can again be either a fixed time t_N or a fixed distance d_N. In contrast to dynamic programming though, it is now possible to set a lower and upper limit for either

- the covered distance during the maneuver (D_{min}, D_{max}) if t_N is given, or
- the duration of the driving maneuver (T_{min}, T_{max}) if d_N is given.

Since a major disadvantage of dynamic programming is the lack of a direct influence on both the maneuver time and the covered distance at the same time. Therefore, this type of constraint provides the opportunity to further extend the applicability of optimum speed profiles.

The optimization problem is

$$\min_{\boldsymbol{p}} J$$

subject to:

$$\boldsymbol{C}\boldsymbol{p} \leq 0$$

$$\boldsymbol{c}(\boldsymbol{p}) \leq 0 \tag{4.36}$$

where the objective function J is the total energy that is drained from the battery during the maneuver, $\boldsymbol{p}$ contains the optimization parameters, $\boldsymbol{C}$ is a constant matrix and $\boldsymbol{c}$ is a vector-valued nonlinear function.

To solve this problem, a gradient-based optimization algorithm is used which is designed to be applied on problems whose objective functions and constraints are both continuous and have continuous first derivatives.[26] However, the efficiency of the electric machine – and therefore the objective function – does not meet this requirement because its first derivative is discontinuous. The optimization method might compute only locally optimal results depending on the initialization of the optimization parameters (see chapter 4.2.1 and 4.2.3 for details). To reduce the dependency on the starting point two different methods were investigated to compute an approximation of original efficiency data. Both methods were compared and it turned out that an approximation using weighted constrained linear least squares is the most efficient method (see chapter 4.2.1). Despite this procedure still does not guarantee globally optimal solutions it is assumed that a starting point x_0 can be found that is close enough to the globally optimal solution. Furthermore, it will be shown in chapter 4.2.3 that the obtained solution is quite similar to the one of dynamic programming, which is globally optimal. The basic idea of this chapter was inspired by Grießler (2011).[27]

[26] Cf. The MathWorks Inc.: MATLAB 2007a documentation, fmincon.

[27] Cf. Grießler L. (2011): pp. 42

4.2.1 Introduction to B-splines

This subchapter shows some basics about B-splines. In many applications B-splines are a reasonable mathematical method to fit and interpolate large data sets. In this work they are used to parameterize the flat system output specified in chapter 2.7.

Let $d = (d_j)$, $d_1 \leq d_2 \leq \ldots \leq d_n$ be any sequence of nodes. B-splines $B_{j,k}(\xi)$ of the order k $(1 \leq k < n)$ are then recursively defined by[28]

$$B_{j,1}(\xi) := \begin{cases} 1 & for\ \xi \in [d_j, d_{j+1})\ and\ d_j \neq d_{j+1} \\ 0 & else, \end{cases} \qquad for\ j = 1, \ldots, n-1,$$

$$B_{j,k}(\xi) = \frac{\xi - d_j}{t_{j+k-1} - t_j} B_{j,k-1}(\xi) - \frac{\xi - d_{j+k}}{t_{j+k} - t_{j+1}} B_{j+1,k-1}(\xi)$$

$$for\ k = 2, \ldots, n-1\ j = 1 \ldots, n-k. \tag{4.37}$$

Note that B-splines are both defined by their order k as well as their knot sequence (d_j). In MATLAB, B-splines can be created using the built-in function *bspline*. However, for the purpose of this work, *bspline* has been modified. Figure 4.12 shows the influence of the knot sequence d on the shape of 3rd order B-splines $B_{1,3}(\xi)$.

Note that the B-splines $B_{1,3}(\xi)$ with the knot sequences $d = (0, 0, 0, 1)$ and $B_{1,3}(\xi)$ with $d = (0, 0, 1, 1)$, respectively could also be represented by $B_{1,3}(\xi)$ and $B_{2,3}(\xi)$ with $d = (0, 0, 0, 1, 1)$.

Furthermore, it must be emphasized that is possible to meet certain boundary conditions by choosing a suitable corresponding knot sequence. For example, selecting $d = (0, 0, 0, 1)$ in Figure 4.12 results in $B_{1,3}(\xi = 0) = 1$, while all other B-splines are exactly zero at $\xi = 0$. It will be shown below that exactly this kind of choice of the knot sequence can be utilized to easily create speed profiles with a specific initial and final speed.

---

[28] Cf. Dahmen, W., Reusken A. (2006): pp. 322.

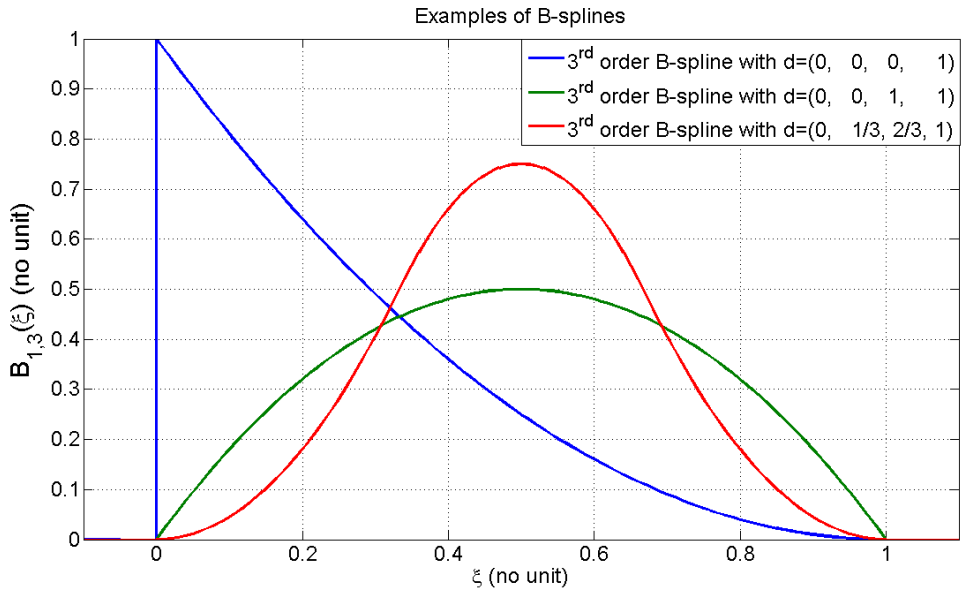

Figure 4.12: Examples of 3rd order B-splines depending on d.

Additional characteristics of B-splines that are important for the implementation below are:

(i) B-splines of the order n are $n-1$ times differentiable,[29]

(ii) Any $B_{j,k}(\xi)$ vanishes outside of the interval $[d_j, d_{j+k})$ and is positive inside that interval.[30]

Any linear combination of B-splines is called a spline, which is defined by

$$v(\xi) = \sum_{j=1}^{n-k} p_j B_{j,k}(\xi), \tag{4.38}$$

where $p_j \in \mathbb{R}$ are the weighting factors for the B-splines.

4.2.2 Electric machine efficiency map fitting

As in any parameter optimization method the optimization parameters have to be somehow initialized. Using a gradient based solver the initial-

[29] Cf. Grießler L. (2011): pp. 42.

[30] Cf. Boor de C. (2001): pp. 91.

ization process becomes critical. It was discovered that by including suboptimal torque splitting to the computation of the combined efficiency data of both electric machines $\eta_{2EM,subopt}$, in many cases the obtained solution is only locally optimal. The reason for this is explained below.

For the sake of a brief notation from now the torque of both electric machines is denoted by $M = M_{2EM}$ and the efficiency is $\eta = \eta(\omega, M) = \eta_{2EM,subopt}$. In Figure 4.13 $\eta(\omega, M)$ is exemplary shown for $\omega = 2500\,rpm$. It is both evident that $\eta(\omega, M)$ is not continuously differentiable with respect to M and that the absolute value of the slope reaches high values near the origin:

$$\frac{\partial \eta(\omega, M)}{\partial M} \text{ ... discontinuous function,}$$

$$\left|\frac{\partial \eta(\omega, M)}{\partial M}\right|_{|M| \ll 1} \gg 1.$$

For this reason two different methods were investigated to resolve the situation by smoothening $\eta(\omega, M)$ for $|M| \ll 1$.

1. A weighted least squares fit of the electric machine efficiency $\eta(\omega, M)$ with the ansatz function $\tilde{\eta} = \alpha_0 + \alpha_{0,1}\omega + \alpha_{1,0}M + \alpha_{1,1}M\omega + \alpha_{1,2}M\omega^2 + \alpha_{2,1}M^2\omega + \alpha_{2,2}M^2\omega^2 + \ ... \ + \alpha_{n,n}M^n\omega^n$, where $n \in \mathbb{Z}^+$. Different results for a range of n between 6 and 12 were investigated.
2. Weighted constrained linear least squares fits at several discrete speeds ω_i using Gaussian ansatz functions with an offset. As a result, several approximated functions for the efficiency at discrete speeds $\eta_i(\omega_i)$ are obtained. The efficiency values for any electric machine speed $\omega_i < \omega < \omega_{i+1}$ are determined by linear interpolation between η_i and η_{i+1}.

Despite both methods reduce the dependency on the starting point, it turned out that the second one is far more effective. Therefore, in this work, method one will not be further considered. Below a brief derivation of weighted least squares fitting is given and the obtained approximation of $\eta(\omega, M)$ will be compared to the original data.

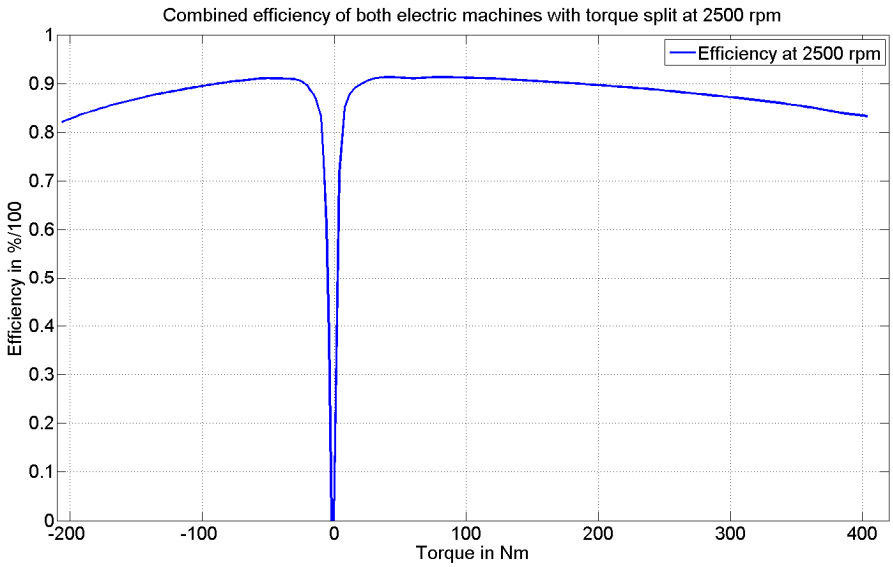

Figure 4.13: Efficiency of both electric machines at 2500 rpm.

As stated above, for an approximation of the efficiency a continuously differentiable and smooth function for values where $|M| \ll 1$ is preferred in order to reduce the dependency of the resulting speed profile on the initial values of the optimization parameters. A function that fits these requirements very well is the Gaussian function.

The first step of the fitting process is to create two sets of functions:

$$\eta_{i+}(M) = \begin{cases} \eta(\omega = \omega_i, M) \; for \; M \geq 0 \\ 0 \qquad\qquad\qquad\quad else \end{cases}$$
$$\eta_{i-}(M) = \begin{cases} \eta(\omega = \omega_i, M) \; for \; M \leq 0 \\ 0 \qquad\qquad\qquad\quad else. \end{cases} \tag{4.39}$$

To scale these functions onto a uniform interval, new arguments are defined as

$$\widetilde{M}_+ = \frac{M}{M_{max}(\omega = \omega_i)}, \quad \widetilde{M}_- = \frac{M}{M_{min}(\omega = \omega_i)}. \tag{4.40}$$

Thus, $\eta_{i\pm}(\widetilde{M}_\pm)$ only need to be fitted inside the interval [0,1]. For simplicity reasons all considerations below refer to only $\eta_{i+}(\widetilde{M}_+)$, whereat the + sign will be dropped for better readability. The ansatz functions used for fitting consist of several displaced Gaussian functions (see Figure 4.14) combined with an offset as stated in equation (4.41).

$$\tilde{\eta}_i = \alpha_0 + \sum_{k=1}^{8} \alpha_{k,i} e^{0.5\left(\frac{\tilde{M}-\mu_k}{\sigma}\right)^2} \tag{4.41}$$

The corresponding expectation values are given by $\mu_k = \{\mu_1, \mu_2, \dots, \mu_8\} = \{0.24, 0.355, \dots, 1.045\}$, the variance is $\sigma = 0.07$ and the scaling factors $\alpha_{k,i} \in \mathbb{R}$ are optimization parameters. Figure 4.14 shows the Gaussian functions as well as their sum (green line), which can be shaped by varying α_k.

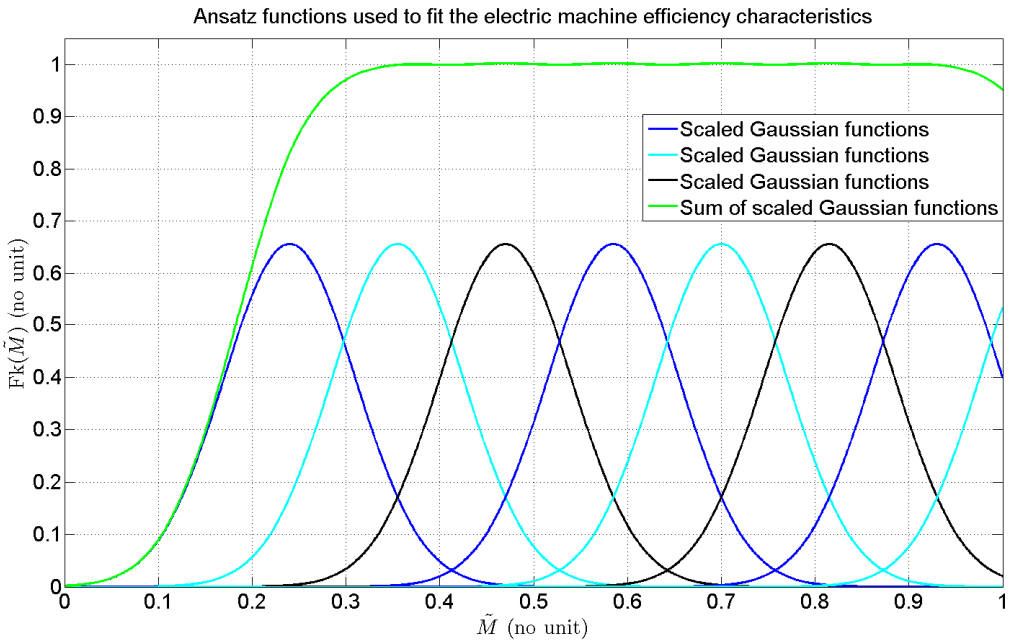

Figure 4.14: Ansatz functions used to approximate the electric machine efficiency.

The fundamental idea now is to determine all $\alpha_{i,k}$ $(k = 1,2, \dots, K), K = 8$ in a way such that

$$\left(w_1 \varepsilon_{i,1}\right)^2 + \left(w_2 \varepsilon_{i,2}\right)^2 + \dots + \left(w_{n-1} \varepsilon_{i,n-1}\right)^2 + \left(w_n \varepsilon_{i,n}\right)^2 \rightarrow min, \tag{4.42}$$

where $w_j \varepsilon_{i,j}$ is the weighted approximation error occurring due to the fitting process at $\eta\left(\omega = \omega_i, M = M_j\right)$.

$$\varepsilon_{i,j} = \eta\left(\omega = \omega_i, M = M_j\right) - \left(\alpha_0 + \sum_{k=1}^{K} \alpha_{i,k} e^{-0.5\left(\frac{\tilde{M}_j-\mu_k}{\sigma}\right)^2}\right)$$

$$= \eta\left(\omega_i, M_j\right) - \tilde{\eta}_i\left(\tilde{M} = \tilde{M}_j\right) = \eta_{i,j} - \tilde{\eta}_{i,j}. \tag{4.43}$$

The distance between the nodes M_j can be specified by the user in a MATLAB m-file and is by default 2 Nm for $M < 0$ and 4 Nm for $M \geq 0$, respectively. For better readability the index i will be omitted from now on. With

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_{\mathrm{i},1} \\ \vdots \\ \varepsilon_{i,n} \end{bmatrix}, \quad \boldsymbol{\eta} = \begin{bmatrix} \eta_{i,1} \\ \vdots \\ \eta_{i,n} \end{bmatrix}, \quad \tilde{\boldsymbol{\alpha}} = \begin{bmatrix} \alpha_0 \\ \boldsymbol{\alpha} \end{bmatrix}, \quad \boldsymbol{\alpha} = \begin{bmatrix} \alpha_{\mathrm{i},1} \\ \vdots \\ \alpha_{i,K} \end{bmatrix},$$

$$\tilde{\boldsymbol{H}} = [\boldsymbol{h} \quad \boldsymbol{H}], \quad \boldsymbol{H} = \begin{bmatrix} e^{0.5\left(\frac{\tilde{M}_1 - \mu_1}{\sigma}\right)^2} & \cdots & e^{0.5\left(\frac{\tilde{M}_1 - \mu_k}{\sigma}\right)^2} \\ \vdots & \ddots & \vdots \\ e^{0.5\left(\frac{\tilde{M}_n - \mu_1}{\sigma}\right)^2} & \cdots & e^{0.5\left(\frac{\tilde{M}_n - \mu_k}{\sigma}\right)^2} \end{bmatrix}, \quad \boldsymbol{h} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

equations (4.43) can be written in vector form as

$$\boldsymbol{\varepsilon} = \boldsymbol{\eta} - \tilde{\boldsymbol{H}}\tilde{\boldsymbol{\alpha}}. \tag{4.44}$$

Note that the partitioning of $\tilde{\boldsymbol{H}}$ into $\boldsymbol{h}$ and $\boldsymbol{H}$ as well as $\tilde{\boldsymbol{\alpha}}$ into $\boldsymbol{\alpha}$ and α_1 will come in handy further below. With $\boldsymbol{W} = diag(w_k^2)$ also equation (4.42) can be stated in a more concise form:

$$\boldsymbol{\varepsilon}^{\boldsymbol{T}}\boldsymbol{W}\boldsymbol{\varepsilon} \rightarrow min. \tag{4.45}$$

The sum of the weighted least squares errors $\boldsymbol{\varepsilon}^{\boldsymbol{T}}\boldsymbol{W}\boldsymbol{\varepsilon}$ can be rewritten as

$$\begin{aligned} \boldsymbol{\varepsilon}^{\boldsymbol{T}}\boldsymbol{W}\boldsymbol{\varepsilon} &= \left(\boldsymbol{\eta} - \tilde{\boldsymbol{H}}\tilde{\boldsymbol{\alpha}}\right)^T \boldsymbol{W}\left(\boldsymbol{\eta} - \tilde{\boldsymbol{H}}\tilde{\boldsymbol{\alpha}}\right) \\ &= \left(\boldsymbol{\eta}^T - \tilde{\boldsymbol{\alpha}}^T\tilde{\boldsymbol{H}}^T\right)\boldsymbol{W}\left(\boldsymbol{\eta} - \tilde{\boldsymbol{H}}\tilde{\boldsymbol{\alpha}}\right) \\ &= \boldsymbol{\eta}^T\mathbf{W}\boldsymbol{\eta} - 2\boldsymbol{\eta}^T\boldsymbol{W}\tilde{\boldsymbol{H}}\tilde{\boldsymbol{\alpha}} + \tilde{\boldsymbol{\alpha}}^T\tilde{\boldsymbol{H}}^T\boldsymbol{W}\tilde{\boldsymbol{H}}\tilde{\boldsymbol{\alpha}}. \end{aligned} \tag{4.46}$$

Note that α_0 is not an optimization parameter. Thus, only $\boldsymbol{\alpha}$ instead of $\tilde{\boldsymbol{\alpha}}$ is to be optimized. The optimization problem is

$$\min_{\boldsymbol{\alpha}}\{\boldsymbol{\varepsilon}^{\boldsymbol{T}}\boldsymbol{W}\boldsymbol{\varepsilon}\} = \min_{\boldsymbol{\alpha}}\left\{\boldsymbol{\eta}^T\mathbf{W}\boldsymbol{\eta} - 2\boldsymbol{\eta}^T\boldsymbol{W}\tilde{\boldsymbol{H}}\tilde{\boldsymbol{\alpha}} + \tilde{\boldsymbol{\alpha}}^T\tilde{\boldsymbol{H}}^T\boldsymbol{W}\tilde{\boldsymbol{H}}\tilde{\boldsymbol{\alpha}}\right\}. \tag{4.47}$$

Rearranging $\boldsymbol{\varepsilon}^{\boldsymbol{T}}\boldsymbol{W}\boldsymbol{\varepsilon}$ gives

$$\begin{aligned} \boldsymbol{\varepsilon}^{\boldsymbol{T}}\boldsymbol{W}\boldsymbol{\varepsilon} &= \boldsymbol{\eta}^T\boldsymbol{W}\boldsymbol{\eta} - 2\boldsymbol{\eta}^T\boldsymbol{W}[\boldsymbol{h} \quad \boldsymbol{H}]\begin{bmatrix} \alpha_0 \\ \boldsymbol{\alpha} \end{bmatrix} + [\alpha_0 \quad \boldsymbol{\alpha}^T]\begin{bmatrix} \boldsymbol{h}^T \\ \boldsymbol{H}^T \end{bmatrix}\boldsymbol{W}[\boldsymbol{h} \quad \boldsymbol{H}]\begin{bmatrix} \alpha_0 \\ \boldsymbol{\alpha} \end{bmatrix} \\ &= \boldsymbol{\eta}^T\mathbf{W}\boldsymbol{\eta} - 2\boldsymbol{\eta}^T\boldsymbol{W}(\boldsymbol{h}\alpha_0 + \boldsymbol{H}\boldsymbol{\alpha}) + (\alpha_0\boldsymbol{h}^T + \boldsymbol{\alpha}^T\boldsymbol{H}^T)\boldsymbol{W}(\boldsymbol{h}\alpha_0 + \boldsymbol{H}\boldsymbol{\alpha}) \\ &= \boldsymbol{\eta}^T\boldsymbol{W}\boldsymbol{\eta} - 2\boldsymbol{\eta}^T\boldsymbol{W}(\boldsymbol{h}\alpha_0 + \boldsymbol{H}\boldsymbol{\alpha}) + \alpha_0^2\boldsymbol{h}^T\boldsymbol{W}\boldsymbol{h} + \\ &\quad 2\alpha_0\boldsymbol{h}^T\boldsymbol{W}\boldsymbol{H}\boldsymbol{\alpha} + \boldsymbol{\alpha}^T\boldsymbol{H}^T\boldsymbol{W}\boldsymbol{H}\boldsymbol{\alpha}, \end{aligned} \tag{4.48}$$

where $\alpha_0 \boldsymbol{h}^T \boldsymbol{W} \boldsymbol{H} \boldsymbol{\alpha} = \boldsymbol{\alpha}^T \boldsymbol{H}^T \boldsymbol{W} \boldsymbol{h} \alpha_0$ and $\boldsymbol{H}^T \boldsymbol{W} \boldsymbol{H}$ is a symmetric matrix, hence $\boldsymbol{H}^T \boldsymbol{W} \boldsymbol{H} = (\boldsymbol{H}^T \boldsymbol{W} \boldsymbol{H})^{\boldsymbol{T}}$. A necessary condition for the minimization is

$$\frac{\partial \boldsymbol{\varepsilon}^{\boldsymbol{T}} \boldsymbol{W} \boldsymbol{\varepsilon}}{\partial \boldsymbol{\alpha}} = 0 = -2(\boldsymbol{\eta}^T \boldsymbol{W} \boldsymbol{H})^T + 2(\alpha_0 \boldsymbol{h}^T \boldsymbol{W} \boldsymbol{H})^T + 2\boldsymbol{H}^T \boldsymbol{W} \boldsymbol{H} \boldsymbol{\alpha} \tag{4.49}$$

Since the columns of $\boldsymbol{H}$ are linearly independent, $\boldsymbol{H}^T \boldsymbol{W} \boldsymbol{H}$ is both an invertible and positive definite matrix. Thus the solution fulfills the sufficient condition

$$\frac{\partial^2 \boldsymbol{\varepsilon}^{\boldsymbol{T}} \boldsymbol{W} \boldsymbol{\varepsilon}}{\partial \boldsymbol{\alpha}^{\boldsymbol{2}}} > 0 \tag{4.50}$$

in order for $\boldsymbol{\varepsilon}^{\boldsymbol{T}} \boldsymbol{W} \boldsymbol{\varepsilon}$ to be a minimum. Equation (4.49) can be written in terms of $\boldsymbol{\alpha}$:[31]

$$\boldsymbol{\alpha} = (\boldsymbol{H}^T \boldsymbol{W} \boldsymbol{H})^{-1} \boldsymbol{H}^T \boldsymbol{W} (\boldsymbol{\eta} - \alpha_0 \boldsymbol{h}). \tag{4.51}$$

Figure 4.15 shows the approximation of the electric machine efficiency at $\omega = 5000\,rpm$. Note that the fitting is *deliberately* bad for $|\widetilde{M}| < 0.24$ because all corresponding weighting factors w_j in equation (4.42) are set to zero. This seems like a huge inaccuracy, but if the dynamic programming algorithm is seen as a benchmark, it is well-known that the nature of optimum speed profiles tends to either prefer a rather high absolute value of M (these operating points are well approximated by the fitted function) or freewheeling.

According to Figure 4.14 it is reasonable to consider only the leftmost base function when analyzing the properties of $\tilde{\eta}_i$ at $\widetilde{M}$ close to zero. Thus, the first derivative of the fitted function can be approximated by the first derivative of the leftmost base function, which can be calculated in MATLAB using *DIFF*. It has the following desired properties:

---

[31] Cf. Hofer A. (2004): lecture notes.

Cf. Bauer R. (2007): pp. 7 - 9.

$$\frac{\partial\tilde{\eta}_i(\tilde{M})}{\partial\tilde{M}} \approx \alpha_1 \frac{\mu_1 - \tilde{M}}{\sigma^2} e^{-0.5\left(\frac{\tilde{M}-\mu_1}{\sigma}\right)^2} < \alpha_1 \frac{3}{\sigma} e^{-\frac{9}{2}}$$
$$for\ \tilde{M} < \mu_1 - 3\sigma. \tag{4.52}$$

Equation (4.52) shows that the value of the first derivative of $\tilde{\eta}_i$ for $\tilde{M} \approx 0$ is sufficiently close to zero. This (empirically) reduces the dependency on the initial values of the optimization parameters.

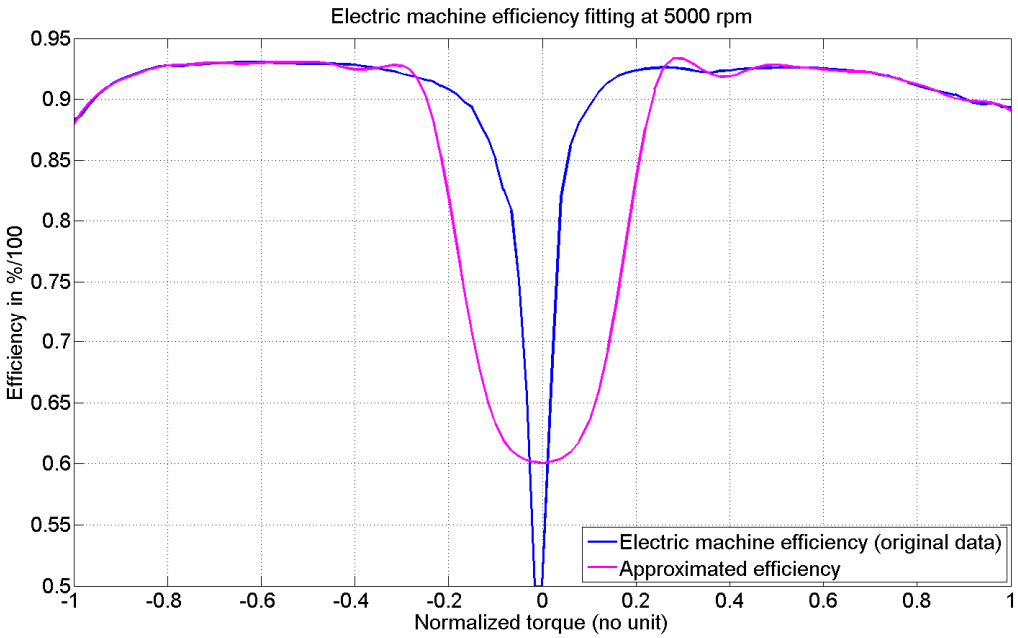

Figure 4.15: Fitted electric machine efficiency at 5000 rpm.

In order to obtain the approximated efficiency for any speed between two nodes ($\omega_i < \omega < \omega_{i+1}$) linear interpolation is applied. Note that due to the electric machine characteristics $M_{max}(\omega_i) \geq M_{max}(\omega_{i+1})$. If $M_{max}(\omega_i) = M_{max}(\omega_{i+1})$ it is straightforward that linear interpolation can always be computed by

$$\tilde{\eta}(\omega, M) = \tilde{\eta}(\omega_i, M) + \frac{\tilde{\eta}\big(\omega_{i+1}, M(\omega)\big) - \tilde{\eta}(\omega_i, M(\omega))}{\omega_{i+1} - \omega_i}(\omega - \omega_i). \tag{4.53}$$

However, in case $M_{max}(\omega_i) > M(\omega) \geq M_{max}(\omega_{i+1})$ or $M_{max}(\omega_i) \geq M(\omega) > M_{max}(\omega_{i+1})$ no exact values for $\tilde{\eta}$ exist. This problem could be solved by extrapolation but in this case the nearest value is used. Thus, equation (4.53) is also valid for $M_{max}(\omega_i) \geq M_{max}(\omega_{i+1})$.

Figure 4.16 shows the absolute value of the error $e = |\eta(\omega, M) - \tilde{\eta}(\omega, M)|$ due to the fitting.

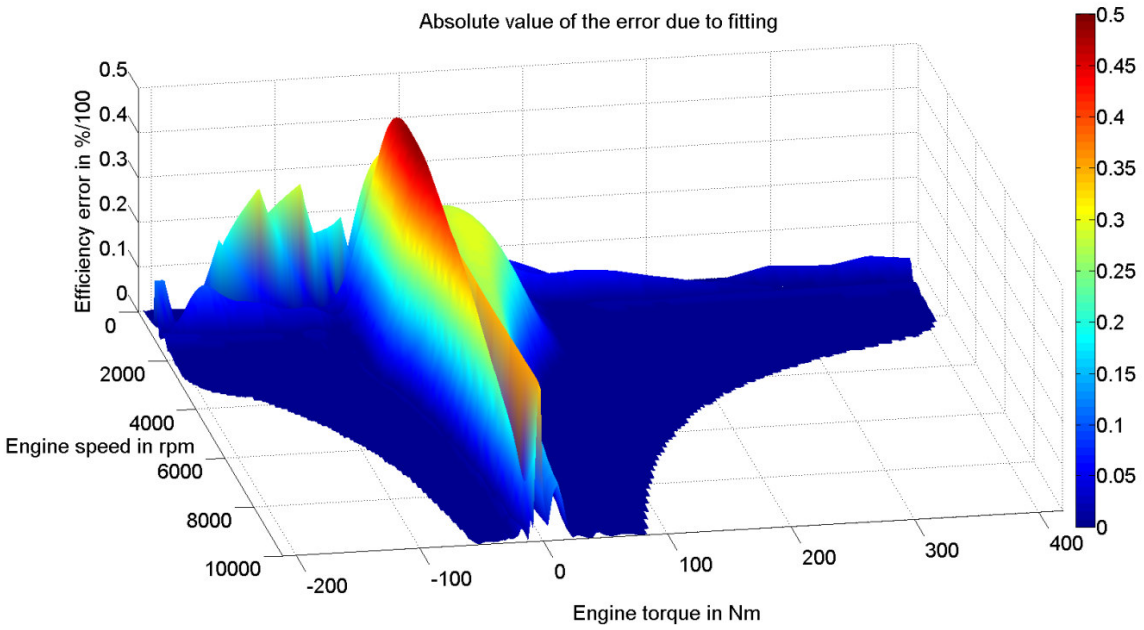

Figure 4.16: Absolute value of the error of the fitted efficiency map

4.2.3 Implementation

In this chapter the implementation of the algorithm will be shown by means of an example. According to equation (4.38) $v(\xi)$ (and therefore the speed profile and the flat system output, respectively) is fully represented by *one* knot sequence d and $n-k$ weighting factors p_j. For the actual velocity profiles the knot sequences are by default constructed as follows (these parameters can be changed in a MATLAB .m-file):

- $d = (0, 0, 0, 50, 100, \ldots, d_f - 50, d_N, d_N, d_N)$ in case the driving maneuver is executed over distance (distance based version of the algorithm). Hereby all values of d are stated in meters and d_N is the traveled distance during the entire driving maneuver. $v(\xi)$ is in this case a function of distance.
- $d = (0, 0, 0, 5, 10, \ldots, t_f - 5, t_f, t_f, t_f)$ if the driving maneuver is executed over a specific time (time based version of the algorithm). In this case, all values in d are specified in seconds and t_f is the

time in which the driving maneuver takes place. $v(\xi)$ is hereby a function of time.

For the computation of optimal speed profiles only B-splines of order $k = 3$ are generated. Figure 4.17 shows the actual B-splines $B_{j,3}$ that are used for the computation of a 200-meter long speed profile ($\xi_N = 200m$). For $p_j = 1$ it is apparent from the figure that $v(\xi)$ is always one between 0 and d_N. As stated above, choosing d in a certain way can facilitate taking into account boundary conditions. Since $v(\xi = 0)$ and $v(\xi = d_N)$ only depend on $B_{1,3}(\xi = 0)$ and $B_{6,3}(\xi = d_N)$, respectively, both can reach *any* desired value by simply setting p_1 and p_6 to that exact number.

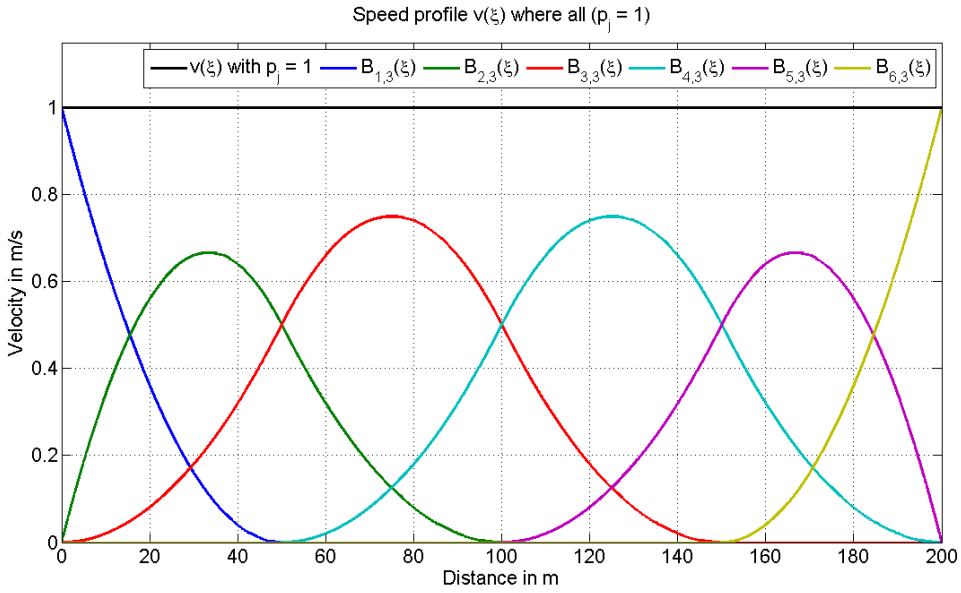

Figure 4.17: B-splines $B_{j,3}(\xi)$ and speed profile $v(\xi)$ if all p_j are one.

Since in the optimization procedure $v(\xi)$ cannot be evaluated on a continuous interval, a discretization has to be carried out. In fact, $v(\xi)$ is being evaluated every $d_s = 2.5m$ and $t_s = 0.25s$, respectively. However, for clarity reasons the example above with the 200-meter long velocity profile will be continued with a fictitious sampling of 25m, which results in only 9 evaluations of $v(\xi)$ at every element of the vector $\boldsymbol{\xi}^T = [0 \quad 25 \quad \cdots \quad 175 \quad 200]$. Note that $\boldsymbol{\xi}$ only depends on the sampling and does not at all correspond to the knot sequence d. If furthermore the B-spline weights are merged into a vector $\boldsymbol{p}^T = [p_1 \quad p_2 \quad \cdots \quad p_{n-k}]$, the discretized spline can be written in vector form:

$$v(\boldsymbol{\xi}) = \begin{bmatrix} v(\xi = 0) \\ v(\xi = 25) \\ \vdots \\ v(\xi = 200) \end{bmatrix} = \boldsymbol{V}\boldsymbol{p}. \tag{4.54}$$

The matrix $\boldsymbol{V}$ is

$$\boldsymbol{V} = \begin{bmatrix} 1.000 & 0 & 0 & 0 & 0 & 0 \\ 0.250 & 0.625 & 0.125 & 0 & 0 & 0 \\ 0 & 0.500 & 0.500 & 0 & 0 & 0 \\ 0 & 0.125 & 0.750 & 0.125 & 0 & 0 \\ 0 & 0 & 0.500 & 0.500 & 0 & 0 \\ 0 & 0 & 0.125 & 0.750 & 0.125 & 0 \\ 0 & 0 & 0 & 0.500 & 0.500 & 0 \\ 0 & 0 & 0 & 0.125 & 0.625 & 0.250 \\ 0 & 0 & 0 & 0 & 0 & 1.000 \end{bmatrix}.$$

The columns j represent the values of $B_{j,3}(\boldsymbol{\xi})$ (note that $B_{j,3}(\boldsymbol{\xi})$ is a column vector) and the lines i correspond to the sum of all B-Splines at the i-th element of $\boldsymbol{\xi}$. It is evident that therefore the sum of every row is exactly one just like $v(\xi)$ is exactly one in Figure 4.17. Furthermore, it is shown that the first and last line in $\boldsymbol{V}$ each have only one element that is one, whereas all other elements are zero. This proofs that any boundary condition for the initial and final vehicle speed can be realized by setting $p_1 = v_0$ and $p_{n-k} = v_f$, respectively.

In the previous chapters the vehicle acceleration was named $\dot{v}$. To emphasize a certain connection between $\dot{v}$ and a matrix $\boldsymbol{A}$, which will be introduced below, it will from now on be called $a(\xi)$. The acceleration can of course be determined by the differentiation of the velocity $a(\xi) = \mathrm{d}v(\xi)/\mathrm{d}\xi$. However, a calculation method similar to equation

(4.54) is preferred because in such a case the acceleration can be directly stated as a function of the optimization parameters p_j. For this purpose a matrix $\boldsymbol{A}$ that fulfills the following equation is desired:

$$a(\boldsymbol{\xi}) = \begin{bmatrix} a(\xi = 0) \\ a(\xi = 25) \\ \vdots \\ a(\xi = 200) \end{bmatrix} = \boldsymbol{A}\boldsymbol{p}. \tag{4.55}$$

In this case the first $n-1=8$ rows of $\boldsymbol{A}$ can simply be generated through $\boldsymbol{V}$ by computing the differences of the i-th and $i+1$-th line and dividing them by the sampling rate d_s. The value for $a(\xi=200)$ cannot be determined because at $\xi=200$ only the final velocity but not the final acceleration is considered in the optimization procedure. Therefore, any arbitrary value for $a(\xi=200)$ can be chosen only to assure that $\boldsymbol{V}$ and $\boldsymbol{A}$ both have the same dimension.

A remark to the correlation between functions of time and distance:

As mentioned above in the optimization algorithm the vehicle speed and the acceleration are considered as functions of ξ which either refers to the time t or the distance d. However, even if $\xi=d$, acceleration constraints or the driving resistance are usually given as functions of the time. Thus, it shall be pointed out, that there exists a clear relation between $a(t)$ and $\tilde{a}(d)$.

The vehicle speed as a function of the distance is defined by $\tilde{v}(d)=v\big(t(d)\big)$. The first derivative with respect to d is

$$\frac{\mathrm{d}\tilde{v}(d)}{\mathrm{d}d}=\tilde{a}(d)=\frac{\partial v(t)}{\partial t}\frac{\mathrm{d}t(d)}{\mathrm{d}d}=a(t)\frac{1}{\tilde{v}(d)}. \tag{4.56}$$

Acceleration constraint:

Since $v(\xi)$ is a flat system output it is possible to compute the actuating variable $M(\xi)$ with the knowledge of p_j. Likewise, constraints for $M(\xi)$ as they occur due to the power limitations of the electric machines, as well as due to comfort reasons can be transformed into constraints for $v(\xi)$ and p_j, respectively. According to equations (2.2) and (2.7) the vehicle acceleration is a function of both the driving resistance and the torque of both electric machines: $a(t)=a\left(M_{res}\big(v(t)\big),M_{EM}(t)\right)$. In this manner the minimum and maximum torque the electric machines can provide at any given velocity can be transformed in $a_{min}\big(v(t)\big)$ and $a_{max}\big(v(t)\big)$. At this point it is important to point out the distinction between $a(t)$ as a

function of the time and $a(d)$ as a function of the distance. All constraints will always be given as a function of time. In case the B-splines are defined as functions of time, of course $a(t) = a(\xi)$. On the other hand, if d denotes the traveled distance, according to equation (4.56) $\tilde{a}(d) = a(t)/\tilde{v}(d)$. Furthermore, let a_{comf} be a constant (positive) acceleration limit that assures traveling comfort. Thus $a_{min}\big(v(\xi)\big) \leq a(t) \leq \min\{a_{max}\big(v(\xi)\big), a_{comf}\}$ must apply. Bringing this inequality to a form that can be processed by the optimization algorithm gives:

$$\begin{aligned} -(\boldsymbol{A}\boldsymbol{p}) \circ (\boldsymbol{V}\boldsymbol{p}) + a_{min}\big(v(\boldsymbol{\xi})\big) &\leq \mathbf{0}, \\ (\boldsymbol{A}\boldsymbol{p}) \circ (\boldsymbol{V}\boldsymbol{p}) - \min\{a_{max}\big(v(\boldsymbol{\xi})\big), a_{comf}\} &\leq \mathbf{0}. \end{aligned} \tag{4.57}$$

Time and distance constraints:

If the velocity profile is defined as a function of the traveled distance, a lower and upper bound can be specified for the time in which the entire maneuver must be carried out. Let d_s be the discretization of $v(\xi)$ and m be the number of evaluations of $v(\xi)$ (m is also the number of elements of $\boldsymbol{\xi}$). With ξ_i being the i-th element of $\boldsymbol{\xi}$ the time t_i it takes to travel the distance d_s at the velocity $v(\xi_i)$ can be approximated by $t_i \approx d_s/v(\xi_i)$. The overall time of the driving maneuver can therefore be estimated by

$$T \approx \sum_{i=1}^{m-1} \frac{d_s}{v(\xi_i)} \tag{4.58}$$

and must neither fall below T_{min} nor exceed T_{max}. With

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_1^T \\ \boldsymbol{v}_2^T \\ \vdots \\ \boldsymbol{v}_m^T \end{bmatrix}$$

a processable syntax for this constraint is:

$$[-1 \quad -1 \quad \cdots \quad -1] \begin{bmatrix} 1/(\boldsymbol{v}_1^T\boldsymbol{p}) \\ 1/(\boldsymbol{v}_2^T\boldsymbol{p}) \\ \vdots \\ 1/(\boldsymbol{v}_{m-1}^T\boldsymbol{p}) \end{bmatrix} d_s + T_{min} \leq 0$$

$$[1 \quad 1 \quad \cdots \quad 1]\begin{bmatrix} 1/(\boldsymbol{v}_1^T\boldsymbol{p}) \\ 1/(\boldsymbol{v}_2^T\boldsymbol{p}) \\ \vdots \\ 1/(\boldsymbol{v}_{m-1}^T\boldsymbol{p}) \end{bmatrix} d_s - T_{max} \leq 0. \tag{4.59}$$

In case the velocity profile is defined as a function of the time the traveled distance can be approximated by

$$D \approx \sum_{i=1}^{m-1} v(\xi_i) t_s. \tag{4.60}$$

With D_{min} and D_{max} being the minimum and maximum distance that the vehicle may travel during the driving maneuver, the distance constraint can be specified by

$$\begin{aligned} [-1 \quad -1 \quad \cdots \quad -1]\boldsymbol{V}\boldsymbol{p} * t_s &\leq -D_{min} \\ [1 \quad 1 \quad \cdots \quad 1]\boldsymbol{V}\boldsymbol{p} * t_s &\leq D_{max}. \end{aligned} \tag{4.61}$$

Velocity constraints:

Due to legal speed limits, the driving speed must not exceed certain limits. In some cases it may also be desirable to not undercut a particular velocity. Therefore it is reasonable to consider velocity constraints at every evaluation of $v(\xi)$. For this purpose let $\boldsymbol{v}_{min}$ and $\boldsymbol{v}_{max}$ be vectors whose i-th elements correspond to the minimum and maximum feasible velocity at the ith element of $\boldsymbol{\xi}$ so that $\boldsymbol{v}_{min} \leq \boldsymbol{V}\boldsymbol{p} \leq \boldsymbol{v}_{max}$. Bringing this to the notation used in the formulation of the optimization problem in (4.36) gives:

$$\begin{aligned} \boldsymbol{V}\boldsymbol{p} &\leq \boldsymbol{v}_{max}, \\ -\boldsymbol{V}\boldsymbol{p} &\leq -\boldsymbol{v}_{min}. \end{aligned} \tag{4.62}$$

Objective function:

The objective function J of the optimization problem in (4.36) is defined as

$$J = \boldsymbol{t}^T \boldsymbol{P}_{el}, \tag{4.63}$$

where the vector $\boldsymbol{t}^T = [t_1 - t_0, \quad t_2 - t_1, \quad \cdots, \quad t_{m-1} - t_{m-2}]$ contains the time intervals between two simulation steps and the elements of $\boldsymbol{P}_{el}$ represent the electrical power at each step:

$$\boldsymbol{P}_{el} = \begin{bmatrix} P_{el}(t_0) \\ \vdots \\ P_{el}(t_{m-2}) \end{bmatrix}. \tag{4.64}$$

In turn, the electrical power at $t = t_i$ consists of the electric machine power as well as the power loss in the battery:

$$P_{el}(t_i) = P_{EM}(t_i) + P_{Batt,loss}(t_i), \tag{4.65}$$

where

$$P_{EM}(t_i) = \begin{cases} \omega(t_i) M(t_i) \tilde{\eta}\big(M(t_i), \omega(t_i)\big) & for\ M(t_i) < 0 \\ \dfrac{\omega(t_i) M(t_i)}{\tilde{\eta}\big(M(t_i), \omega(t_i)\big)} & for\ M(t_i) \geq 0, \end{cases} \tag{4.66}$$

and

$$P_{Batt,loss}(t_i) = I^2(t_i) R. \tag{4.67}$$

Hereby, R denotes the internal battery resistance and $I(t_i)$ is the current that is drained from the battery, which is given by

$$I(t_i) = \frac{P_{EM}(t_i)}{V}. \tag{4.68}$$

4.2.4 Results and discussion

It is an interesting aspect to compare the results obtained by using the dynamic programming algorithm and the approach using B-splines. For this purpose two use cases are explained below. In example 1 the exact same boundary conditions are used as in the DP example in chapter 4.1.2 to point out some similarities and differences of the result. Example 2 shows that – compared to DP – additional use cases can be covered by the algorithm based on the B-splines parameter optimization.

Example 1:

In order to compare the results with the ones obtained with dynamic programming consider the same example as above where the boundary conditions are given by $v(\xi_0) = 100\,km/h$ and $v(\xi_N = 500m) = 50\,km/h$. The B-splines parameter vector is initialized by

$$\boldsymbol{p}^T = \left[v(\xi_0) \quad \frac{v(\xi_0) + v(\xi_N)}{2} \quad \dots \quad \frac{v(\xi_0) + v(\xi_N)}{2} \quad v(\xi_N)\right]. \tag{4.69}$$

Figure 4.18 clearly indicates that the velocity trajectory as well as the optimal control inputs are similar to those of DP. Note that the derivatives of third-order B-splines are piecewise linear functions. Thus, sudden changes of the vehicle acceleration are not possible. This can be observed in Figure 4.18 where at a maneuver time of about 12 seconds the torque becomes zero relatively slow which results in a smooth transition from regenerative breaking to freewheeling. An important characteristic of the velocity trajectories computed with B-splines is that they are quite smooth which greatly benefits driving comfort. Therefore – in contrast to trajectories computed with DP – no filtering is necessary.

Another advantage of using a parameter optimization method is that additional constraints can be applied. In the example above it is possible to specify a minimum and/or maximum maneuver time. In the DP algorithm this is not possible.

A major drawback is that in no case a globally optimal solution is guaranteed. In fact there is a number of scenarios where the solution is highly dependent on the initialization of the B-spline parameter vector $\boldsymbol{p}$. A smart choice of the initial values can only reduce the risk of obtaining a locally optimal solution. Dynamic programming is therefore an essential tool for benchmarking. Furthermore, dynamic programming is extremely robust against disturbances and simultaneously computes a huge set of optimal policies. Thus, in contrast to the parameter optimization method it is not necessary to recalculate the optimal trajectory if the vehicle unintentionally significantly differs from the optimal velocity

profile. Evidentially, it turns out that in some cases the computational effort of DP is far less.

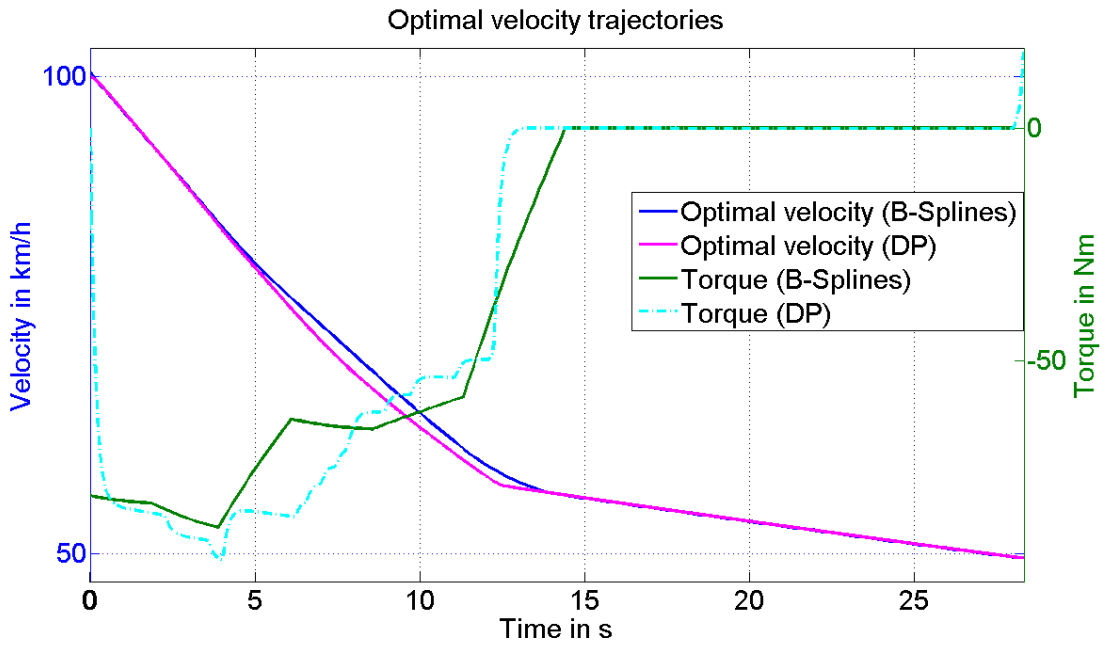

Figure 4.18: Comparison of velocity profiles computed by using B-splines and dynamic programming.

Nevertheless the author proposes to further investigate using B-splines in the optimization procedure because constraints for D_{min}, D_{max} and T_{min}, T_{max}, respectively may prove extremely useful for a broad variety of use cases (one is shown below in example 2). Further studies should mainly focus on eliminating or at least decreasing the dependency on the initial values of the optimization variables.

Example 2:

Since traffic lights play an essential role in everyday traffic scenarios, it is now emphasized that it also makes sense to apply the algorithm presented above in use cases where traffic light signals must be considered.

Assume that a motorist is traveling at 50km/h (13.$\dot{8}$m/s). If the velocity stayed unchanged within the next 40 seconds the driver would travel 555.$\dot{5}$m. However, 450m in front of the vehicle there is a red light that

will turn green in exactly 40 seconds. Hence, an optimization problem can be formulated with $v(\xi_0 = 0) = v(\xi_N = 40s) = 50km/h$ and $D_{max} = 450m$. In addition the motorist wants to travel at least $D_{min} = 430m$. In this case it is assumed that the traveled distance will then be exactly 430m since it seems plausible that less energy is required to travel a shorter distance. Furthermore it is desired not to decelerate below $v_{min} = 30km/h$. To ensure driving comfort the acceleration is limited to $a_{max} = 1.25m/s$.

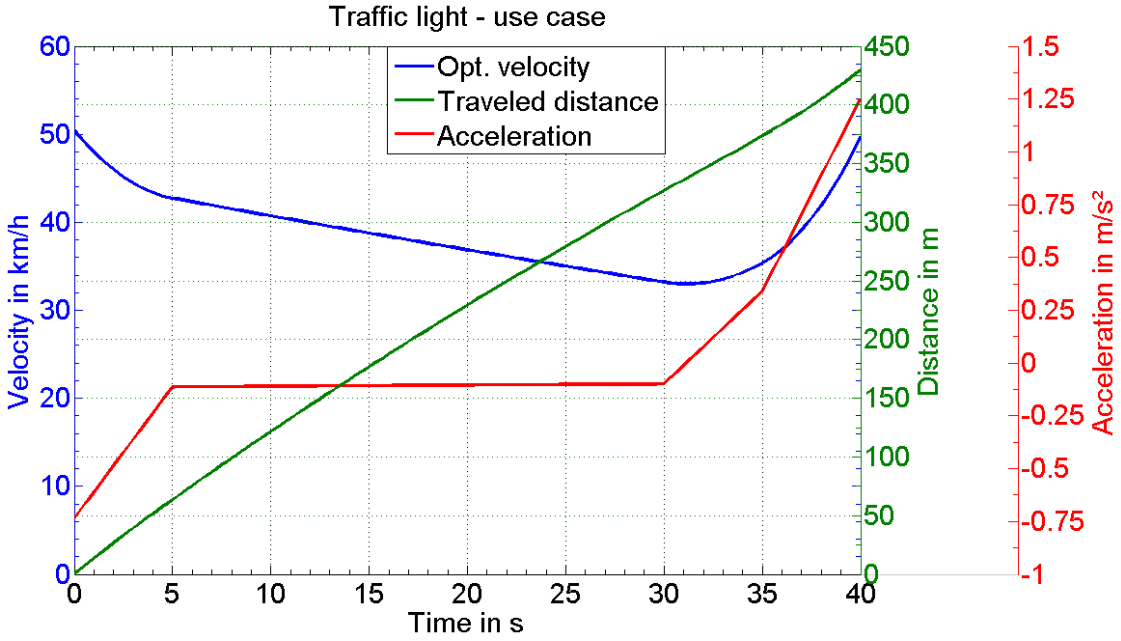

Figure 4.19: Energy-optimal velocity trajectory for approaching a traffic light.

Figure 4.19 shows the energy-optimal velocity trajectory to approach the traffic light. During the first five seconds energy is recuperated. Then the vehicle freewheels and finally accelerates up to 50km/h. Note that there is a strong similarity to the deceleration use cases explained above. As expected the covered distance is $D_{min} = 430m$. Also, the constraints for v_{min} and a_{max}, respectively are met.

5 A concept study for an advanced route planner

Car navigation systems detect the supposedly best route to a destination. They support the driver and some tasks as for example reading a map become obsolete. A typical car navigation system provides several different settings regarding the criteria to determine the optimal route or cost function. Normally the driver can choose between computing a fastest or shortest route and specify preferences to avoid motorways, toll roads and ferries. Another important aspect is that a route planner should take into account real-time traffic conditions such as the presence of traffic jams or road works on the current route.[32]

At present, route planners are unable to take into account the estimated energy consumption on a specific route. This, however, is an important and interesting aspect for all-electric vehicles, especially if their range is fairly limited. A driver may want to know if the destination can likely be reached without the need to recharge or change the battery. In addition – besides fastest and shortest routes – drivers might also prefer energy-optimal routes.

This being said, the aim of this chapter is to develop a concept study for a route planner that computes optimal routes according to individual preferences of the driver. This includes the estimation of the energy consumption of a fully electric vehicle by using real-time traffic information provided by c2i communication and 3D GPS maps to consider road elevation profiles. It is emphasized that this chapter only describes a concept study and that some functionalities cannot yet be fully applied in realistic simulations.

Many algorithms exist that can solve routing problems. For this concept study the Bellman-Ford algorithm was chosen. It is implemented in both a MATLAB m-file and in Simulink using an Embedded MATLAB

[32] Cf. Flinsenberg I. (2004): pp. 1, 4.

Function to enable co-simulations with IPG CarMaker and AVL CRUISE.

5.1 The Bellman-Ford algorithm

Consider a set of G cities. Some of them are linked by direct roads. If two cities are directly linked the cost to travel from city i to city j is denoted by r_{ij} which represents a weighted sum of the travel time, the distance and the expected energy consumption. The goal is to find the optimal path between cities that minimizes the sum of all costs. From this general problem formulation it is evident that the solution can also be obtained by using a dynamic programming algorithm.

The Bellman-Ford algorithm is indeed quite similar to dynamic programming and is also based on Bellman's principle of optimality. However, a significant difference is that instead of calculating the costs-to-go the optimal costs from one particular city to all other cities and the corresponding optimal decisions can be determined. Therefore, the optimal path to any city can be calculated. If the destination changes but the starting point remains the same, the optimal costs in the road network do not need to be recalculated. Note that this is the exact opposite to dynamic programming.[33]

Figure 5.1 shows an example of a road network. The letters A-G represent seven different cities. Without loss of generality assume that A is the starting point. The optimal cost to come to every city starting from A is denoted by $H_j,\ j = B, C, \dots, G$. Since also the amount of energy to travel between cities is considered it is evident that due to regenerative breaking the cost to travel between cities r_{ij} could become negative. The Bellman-Ford algorithm can – in contrast to the faster Dijkstra algorithm – handle negative costs. However, a detection of negative cycles must be implemented. A negative cycle is a path that can be repeatedly followed

[33] Cf. Bellman R. (1957): pp. 1 - 2.

and whose costs sum up to an arbitrarily low negative number. Fortunately it is not possible to recuperate an infinite amount of energy so that in this case negative cycles do not need to be considered.

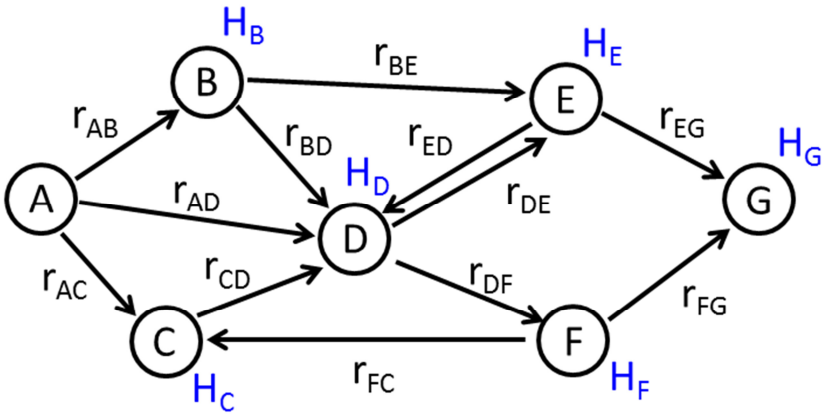

Figure 5.1: Example of a simple road network (1).

Adapted from: Humboldt-Universität zu Berlin, 5/20/2012.

The road network is fully represented by a matrix

$$\boldsymbol{R} = \begin{bmatrix} \infty & r_{AB} & r_{AC} & r_{AD} & \infty & \infty & \infty \\ \infty & \infty & \infty & r_{BD} & r_{BE} & \infty & \infty \\ \infty & \infty & \infty & r_{CD} & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & r_{DE} & r_{DF} & \infty \\ \infty & \infty & \infty & r_{ED} & \infty & \infty & r_{EG} \\ \infty & \infty & r_{FC} & \infty & \infty & \infty & r_{FG} \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{bmatrix}. \tag{5.1}$$

Every row represents the cost from a particular city to all other cities. Accordingly the columns correspond to the cost to a specific city from every other city. If two cities i and j have no direct connection the cost r_{ij} is set to infinity. If the optimum route is to be planned from a different starting point than A, the elements in $\boldsymbol{R}$ can be rearranged accordingly. The cost to every city starting from A is then calculated by the iterative scheme:

$$\begin{aligned} h_j^{k+1} = \min\{h_i^k + r_{ij}\}, \quad & i = \mathrm{A}, \mathrm{B}, \dots, G, \\ & j = B, C, \dots, G, \\ & h^0 = \{h_A^0, h_B^0, \dots, h_G^0\} = \{0, r_{AB}, \dots, r_{AG}\}, \\ & h_A^k = 0, \forall k. \end{aligned} \tag{5.2}$$

In any case the inequality $H_j \leq h_j^{k+1} \leq h_j^k$ is fulfilled. The sequence will converge after a limited number of steps to H_j. The stop criterion is $h_j^k = h_j^{k+1}, \forall j$. In this case the index i must be stored for all j. This is essential because on the optimal path any city j is reached through the city i. Hence, the optimal route can easily be computed.

In a programming environment the iterative scheme of equation (5.2) can be executed in an inner for-loop for the index i and an outer loop for j. In this way the algorithm is implemented in an Embedded MATLAB Function. To reduce the computational effort in MATLAB .m-files it is also possible to implement equation (5.2) in just one for loop. Let $\boldsymbol{\Lambda}^k$ be a matrix that is defined as

$$\boldsymbol{\Lambda}^k = \begin{bmatrix} h_A^k + r_{AB} & h_A^k + r_{AC} & \dots & h_A^k + r_{AG} \\ h_B^k + r_{BB} & h_B^k + r_{BC} & \dots & h_B^k + r_{BG} \\ \vdots & \vdots & \ddots & \vdots \\ h_G^k + r_{GB} & h_G^k + r_{GC} & \dots & h_G^k + r_{GG} \end{bmatrix}. \tag{5.3}$$

In MATLAB the minimum costs h_j^k as well as the corresponding indices i can be determined by

$$[\boldsymbol{h}^{k+1}\ \boldsymbol{i}^{k+1}] = min(\boldsymbol{\Lambda}^k), \tag{5.4}$$

where $\boldsymbol{h}^k = [h_B^k \quad \dots \quad h_G^k]^T$ and $\boldsymbol{i}^k = [i_B^k \quad \dots \quad i_G^k]^T$.

The stop criterion is $\boldsymbol{h}^k = \boldsymbol{h}^{k+1}$. Subsequently, the optimal path is computed backwards from the destination to the starting point by evaluating the information stored in $\boldsymbol{i}^k$.[34]

Example:

Consider Figure 5.2 which depicts the road network explained above with exemplary values for the costs to travel between cities. The corresponding matrix is

[34] Cf. Gocheva-Ilieva S.: http://evlm.stuba.sk, 5/7/2012.

$$\boldsymbol{R} = \begin{bmatrix} \infty & 3 & -1 & 5 & \infty & \infty & \infty \\ \infty & \infty & \infty & 1 & -2 & \infty & \infty \\ \infty & \infty & \infty & 8 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 4 & 1 & \infty \\ \infty & \infty & \infty & 1 & \infty & \infty & 7 \\ \infty & \infty & 3 & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{bmatrix}. \tag{5.5}$$

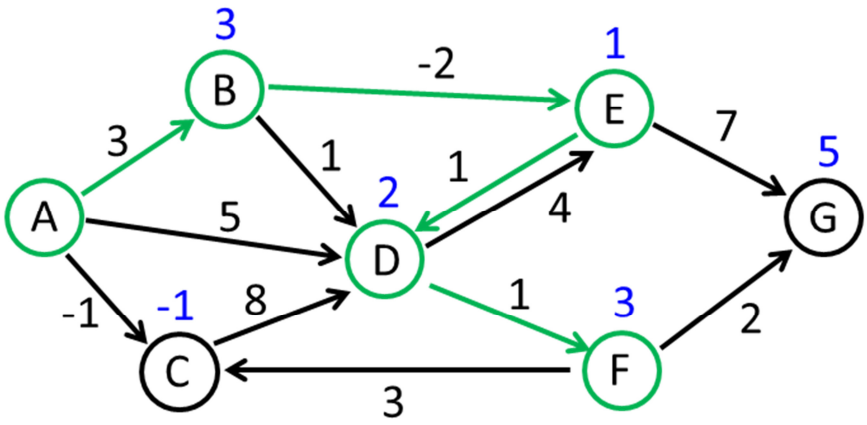

Figure 5.2: Example of a simple road network (2).

There is convergence after a few steps. The result is $\boldsymbol{h}^{k=4} = \boldsymbol{h}^{k=3} = [3 \quad -1 \quad 2 \quad 1 \quad 3 \quad 5]^T$ and $\boldsymbol{i}^{k=3} = [A \quad A \quad E \quad B \quad D \quad F]^T$. Recall that this result shows how to optimally reach every city starting from A. The l-th element corresponds to the j-th city where $l = 1,2,\dots,6$ and $j = B, C, \dots, G$. Now assume that the driver wants to travel to F. According to the fifth element of $\boldsymbol{i}^{k=3}$ it is optimal to reach F via D. The stopover on the optimal route before D is B. Finally, B is directly reached from the starting point. Therefore, once $\boldsymbol{h}^k = \boldsymbol{h}^{k+1}$ and the corresponding $\boldsymbol{i}^k$ have been computed the optimal route to any destination can be determined with very little computational effort. In this case, the optimal route is $A - B - E - D - F$ and the total cost is $H_F = 3$.

5.2 Implementation and discussion

An essential part of the route planner is the computation of the road matrix $\boldsymbol{R}$. On the one hand the costs to travel between cities can change due to real-time traffic conditions. On the other hand the costs must factor in various aspects because it depends on the driver's preferences

whether an energy-optimal route, a time-optimal route or one that minimizes the covered distance is computed. For this purpose the following raw data must be supplied:

- $\boldsymbol{D}$ contains information about the distance between cities.
- $\overline{\boldsymbol{V}}$ is a matrix that has the same structure as $\boldsymbol{R}$ but its elements contain up-to-date average velocities for every road segment. This information may be provided by c2i communication and is required to calculate the travel times $\boldsymbol{T} = \boldsymbol{D} \oslash \overline{\boldsymbol{V}}$.
- $\boldsymbol{\Gamma}_{trf}$ represents the current traffic conditions (e.g. light or heavy traffic, stop-and-go traffic etc.).
- $\boldsymbol{\Gamma}_{inc}$ holds detailed information about the elevation profile of the road map. In some cases this might have a substantial influence on the vehicle's energy consumption.

The road matrix is thereby calculated by

$$\mathbf{R} = w_T \boldsymbol{T} + w_D \boldsymbol{D} + w_E g\left(\boldsymbol{\Gamma}_{trf}, \boldsymbol{\Gamma}_{inc}\right), \tag{5.6}$$

where

$g\left(\boldsymbol{\Gamma}_{trf}, \boldsymbol{\Gamma}_{inc}\right)$ is a scalar function that computes the expected energy consumption for each road segment. The specific vehicle topology may also be taken into account.

w_T is a weighting factor for the travel time between cities,

w_D affects the impact of the traveled distance and

w_E weights the energy consumption.

In case $w_T = 1$ there is an easy interpretation of the weighting factors. The choice $w_D = {}^{1}/_{\kappa_D}$ means that every κ_D meters a penalty of one second is added to the cost function whereas for $w_E = {}^{1}/_{\kappa_E}$ one second is added every κ_E joule.

Note that of course one could argue that one of the weighting factors is redundant. However it is considered as being more intuitive for drivers to choose values for all three.

It is again emphasized that this work only describes a concept study. It is part of this thesis to develop an algorithm that factors in real-time traffic data as well as the driver's preferences to compute an optimal route. It is, however, neither subject of this work to provide such data (i.e. data for $\mathbf{\Gamma}_{trf}$ or $\mathbf{\Gamma}_{inc}$) nor to determine a function g as described in equation (5.6). These functionalities are stated for being implemented at a later time. Nevertheless, in a simple example below it is shown that real-time traffic information can already be processed and time-optimal routes can be computed.

The route planner has been implemented in Simulink to enable future co-simulations with both IPG CarMaker and AVL CRUISE. AVL has demonstrated the potential of the algorithm in a concept simulation. Due to current CarMaker interface limitations the Bellman-Ford algorithm ran isolated in Simulink and the results were exported manually to CarMaker. A road network containing 15 nodes was created in CarMaker (see Figure 5.3).

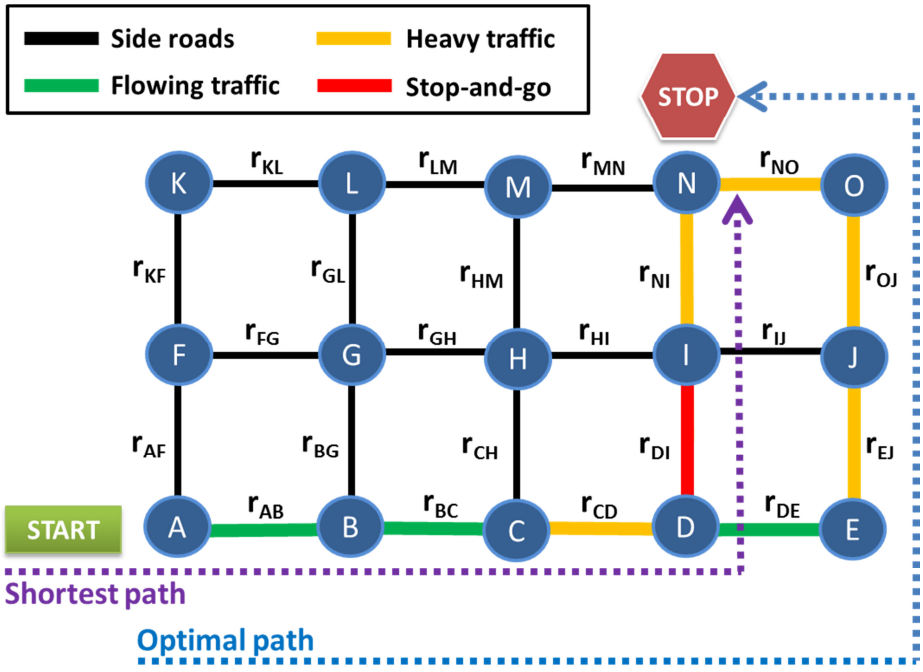

Figure 5.3: Road network in CarMaker.

Adapted from Jones S. et al. (2012): pp. 7-8.

It is desired to travel from the starting point A to the destination N. Black lines between nodes represent small side roads that the driver prefers to avoid. Green lines correspond to road segments where the current average speed is high. On yellow segments there is heavy traffic and the average speed is expected to be less than 50km/h. A red segment indicates a traffic jam or stop-and-go traffic.

Information about the road network topology as well as traffic information is fed to Simulink to determine the fastest route. Accordingly, the weights in equation (5.6) which specify the tradeoff between the fastest route, the shortest route and the energy-optimal route are set to $w_T = 1$ and $w_D = w_E = 0$, respectively.

It is assumed that a driver who has no access to real-time traffic information would drive along the shortest path (A-B-C-D-I-N). The Bellman-Ford algorithm, however, computes the optimal (in terms of travel time) but longer path (A-B-C-D-E-J-O-N). In Figure 5.4 the vehicle velocity is shown for both cases. The letters indicate the nodes of the road network in regard to both paths shown in Figure 5.3.

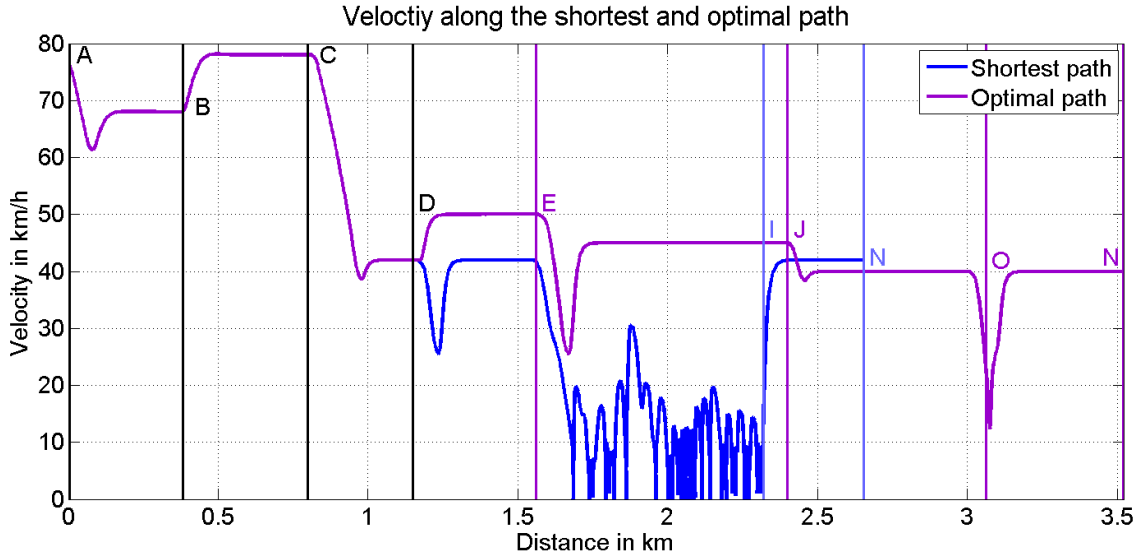

Figure 5.4: Velocity along the shortest route and the optimal route.
Adapted from Jones S. et al. (2012): pp. 8.

The energy consumption along both paths is calculated in a co-simulation of CarMaker and a detailed AVL CRUISE model of the proto-

types. Due to time-optimal route planning the overall energy consumption is reduced by 10.3% compared to a driver that has no access to real-time traffic information and always chooses the shortest route. Simultaneously, the journey time was reduced from 454 seconds to 272 seconds.[35]

[35] Cf. Jones S. et al. (2012): pp. 7 - 8.

6 Conclusion

Given a specific battery capacity there is a broad variety of possibilities to extend the driving range of fully electric vehicles. Some of those strategies are investigated in this work. It is emphasized that the optimization methods complement each other by covering different time horizons. It is demonstrated that dynamic torque splitting saves a significant amount of energy. Therefore torque splitting is also added to the computation of optimal velocity trajectories and is considered in the energy estimation of the route planner.

The most extensive chapter elaborates the determination of optimum velocity trajectories using two different methods. Their strengths and weaknesses are discussed. In addition it is shown that the dynamic programming algorithm is able to realistically calculate the energy consumption in specific driving maneuvers. A comparison of the parameter optimization method with the dynamic programming algorithm showed close similarities in the obtained velocity profile and the optimal control inputs, respectively. However, the choice of the initial values of the B-splines parameters is a crucial factor. Improving the method using B-splines is essential to reduce the dependency on the initialization of the parameter vector in order to ensure applicability in a broader variety of problems. Finally, a concept study of a route planner is shown. Preliminary co-simulations with IGP CarMaker and AVL CRUISE successfully showed that the implementation in the tool chain is possible. As usual in concept studies, much further development is required.

In order to get any optimization method ready to go into mass production it is not sufficient to just cover technical issues. Besides optimizing the energy consumption other aspects as the travel time or driving comfort are extremely important to promote driver's acceptance. Those points are considered in the proposed optimization strategies.

It is envisioned in the OpEneR project that in the future various energy-optimization algorithms will merge into a highly advanced energy manager. For this purpose vehicles must be equipped with a number of

components as for example ACC or an HMI. Furthermore, extensions to the current road infrastructure are badly required to use the full potential of advanced energy management methods.

A Appendix – Vehicle data

New concepts and technologies that are developed within the line of the OpEneR project are demonstrated in two fully electric vehicles which are prototypes and are based on the Peugeot 3008 HYbrid4. In this appendix a brief overview of some key components and additional information about OpEneR is given.

The powertrain consists of two identical synchronous machines from Bosch (model: PSM151-319) that each power one axle. The maximum power of one electric machine is about 57.6kW if the supply voltage is 305V. Both EM can reach a maximum speed of 10,000rpm. Each EM is directly connected to a dog clutch which in turn is connected to the transmission. The transmission ratio is 7.5 and the combined average efficiency of both the transmission and the differential is approximately 92%. This enables the prototypes to reach their maximum velocity of about 161km/h. Driving stability for regenerative breaking is ensured by the ESP®hev. Up to a deceleration of 1.25m/s² the prototypes can use 100% of the recoverable braking energy to charge their batteries.

The lithium ion battery consists of four modules that each have 24 cells. Its nominal idle voltage is approximately 308V depending on the SOC and the temperature. The usable storage capacity is 36.8kW/h, which enables the vehicle to travel 190-250km. The battery mass is about 440kg, which is the main reason why – although the combustion engine has been removed – the overall mass of one prototype is roughly 180kg higher than the one of the Hybrid it is based on. The mass is therefore about 1950kg. Together both electric machines provide up to 410Nm when pulling off from standstill.

OpEneR is a three-year European research project that has launched in May 2011 and is part of the 7th EU Framework Programme (grant agreement n. 285526). It has an overall budget of 7.741.705€ and is funded with 4.400.000€ by the European Union.

Bibliography

OpEneR Project website: http://www.fp7-opener.eu, 4/27/2012.

Back M. (2005): Prädiktive Antriebsregelung zum energieoptimalen Betrieb von Hybridfahrzeugen, Dissertation, Universität Fridericiana Karlsruhe, Universitätsverlag Karlsruhe.

Bellman R. (1954): The Theory of Dynamic Programming, The RAND Corporation.

Bellman R. (1957): On a Routing Problem, The RAND Corporation.

Boor de C. (2001): A Practical Guide to Splines, revised edition, in: Applied Mathematical Sciences, Vol. 27., Springer-Verlag, New York.

Bertsekas D., (2005): Dynamic Programming and Optimal Control, Vol. 1, 3rd Edition, Athena Scientific, Belmont.

Dahmen, W. / Reusken, A. (2006): Numerik für Ingenieure und Naturwissenschaftler, 1st Edition, Springer-Verlag, Berlin.

Flinsenberg I. (2004): Route Planning Algorithms for Car Navigation, dissertation, Technische Universiteit Eindhoven.

Kiencke U., Nielsen L. (2000): Automotive Control Systems: For Engine, Driveline, and Vehicle, 1st Edition, Springer-Verlag.

Kirk D. (2004): Optimal Control Theory, An Introduction, Dover Edition, Mineola.

Gausch, F. (2010): Mehrgrößensysteme, lecture notes, Graz University of Technology.

Gocheva-Illeva S.: The Bellman-Ford Algorithm, http://evlm.stuba.sk, 5/7/2012.

Guzzella L. Sciarretta A. (2007): Vehicle Propulsion Systems, Introduction to Modeling and Optimization, 2nd Edition, Springer-Verlag, Berlin.

Grießler L. (2011): Fahrstrategieoptimierung bei Nutzfahrzeugen mit Hilfe vorausschauender Information, Master's thesis, Johannes Kepler University Linz.

Hofer A. (2004): Computergestützte Modellbildung und Simulation, Lecture notes, Graz University of Technology.

Humboldt-Universität zu Berlin, Institut für Informatik, http://www.informatik.hu-berlin.de, 5/20/2012.

Jones S. et al. (2012): Optimal Fully Electric Vehicle Recovery in an Intelligent Transportation System, ITS World Congress, Vienna.

Bauer R. (2007): Zustandsschätzung Filterung, Version 1.0, lecture notes, Graz University of Technology.

The MathWorks Inc.: MATLAB 2007a documentation, fmincon.

Robert Bosch GmbH: http://www.bosch-presse.de/presseforum/details.htm?txtID=5291, 7/18/2012.

Steinmann Jochen: Chassis Systems Control (CC/ENA3), Robert Bosch GmbH, Heilbronn: e-mail, 2/9/2012.

Zeitz M. (2010): Differentielle Flachheit: Eine nützliche Methodik auch für lineare SISO-Systeme, at – Automatisierungstechnik: Vol. 58, No. 1, pp. 5-13.