Masterarbeit

Optimale Betriebsstrategie eines Seriellen Hybridfahrzeugs basierend auf Modell-Prädiktion

vorgelegt von

Gabor Pongracz

Institut für Regelungs- und Automatisierungstechnik,
Technische Universität Graz
A-8010 Graz

Graz, November 2011

Begutachter: Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. tit.Univ.-Prof. Anton Hofer, TU Graz
Betreuer: Dipl.-Ing. Martin Benedikt, ViF

Kurzfassung

Diese Arbeit befasst sich mit der Bestimmung einer optimalen Betriebsstrategie bezüglich des Kraftstoffverbrauchs mit Hilfe der dynamischen Programmierung für ein serielles Hybridfahrzeug basierend auf Modellprädiktion.

Das Fahrzeugmodell wird basierend auf dem quasistatischen Ansatz unter MATLAB® mit der ausschließlichen Verwendung von Matrizenoperationen formuliert. Dieses Modell wird von einem Algorithmus für die dynamische Programmierung verwendet, um eine optimale Steuergröße für die Verbrennungskraftmaschine des Range Extenders zu berechnen.

Ein geschlossener Regelkreis wird mittels modellprädiktiver Regelung erstellt, damit Störungen, die unter anderem durch die Verwendung eines prädizierten Geschwindigkeitsprofils entstehen, entgegengewirkt wird. Um die Ausnützung des Zustandsraums durch den modellprädiktiven Regler nicht einzuschränken, wird eine Methode zur Vorberechnung der Endladungswerte der Batterie zu jeder Periode des Prädiktionshorizonts vorgestellt.

Die Visualisierung erfolgt durch Darstellung der Ergebnisse in Google Maps, womit mehrere Szenarien evaluiert werden.

Abstract

This paper addresses the problem of creating an optimal control strategy regarding fuel consumption for a serial hybrid vehicle based on dynamic programming and model prediction.

The vehicle model is formulated in MATLAB® based on the quasi static approach using matrix computations only. This model is subsequently used with a dynamic programming algorithm to compute an optimal open-loop control signal which minimizes fuel consumption of the range extender.

A closed-loop controller is achieved using the model predictive control method in order to compensate disturbances in the predicted vehicle speed vector. In order to avoid loss of utilization of the state space, a method is introduced which predicts the ending state of the charge for each model predictive control period.

A visualization of the results is achieved by plotting the resulting data in Google Maps. With this visualization, different scenarios are evaluated.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Ort	Datum	Unterschrift

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.

Place	Date	Signature

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Danksagung

Mein Dank gilt Herrn Prof. Anton Hofer vom Institut für Regelungs- und Automatisierungstechnik der Technischen Universität Graz für seine Unterstützung in allen fachlichen Fragen, für die konstruktiven Verbesserungsvorschläge dieser Arbeit betreffend und für sein Engagement während den zahlreichen Vorlesungen mit denen er mein Studium bereichert und mich fachlich wie auch persönlich kontinuierlich motiviert hat.

Weiters möchte ich Herrn Dipl.-Ing. Martin Benedikt und dem Kompetenzzentrum *Das virtuelle Fahrzeug* Forschungssgesellschaft mbH. für die Ermöglichung dieser Masterarbeit und für die dafür gebotene Unterstützung danken.

Zu guter Letzt gilt mein Dank meiner Familie, meiner Freundin und all meinen Freunden, die mir in jeder Lebenslage beigestanden sind.

Gabor Pongracz
31. Oktober 2011, Graz

Kapitel 1

Einführung

Durch den kontinuierlich steigenden Bedarf und gleichzeitigem Mangel an fossilen Brennstoffen sind in den letzten Jahren seitens der Fahrzeugindustrie große Anstrengungen unternommen worden, um alternative Antriebsarten vorzustellen. Dabei ist ein Fokus auf den verschiedenen Elektrifizierungsstufen des Antriebstrangs und deren Versorgung mit notwendiger Energie.

Ein Vertreter dieser neuen, elektrifizierten Antriebsarchitektur ist der Serienhybrid. Dabei ist für den Antrieb allein eine elektrische Maschine zuständig, die über eine Batterie mit elektrischem Strom versorgt wird und in geeigneten Betriebszuständen Energie in diese zurückspeisen kann. Geht die gespeicherte Energie in der Batterie zur Neige, wird ein sogenannter Range Extender, bestehend aus dem Verbund eines elektrischen Generators und einer Verbrennungskraftmaschine, in Betrieb genommen, welcher die Batterie mit zusätzlichem elektrischen Strom versorgt.

Ziel dieser Arbeit ist es, für ein Fahrzeug dieses Antriebtyps eine optimale Betriebsstrategie bezüglich des Kraftstoffverbrauchs für den Betrieb des Range Extenders zu finden. Dabei gilt die Nebenbedingung, dass am Ende der Fahrt die Batterie den gleichen Ladungszustand wie am Anfang aufweisen soll, ansonsten hätte man Kraftstoff auf Kosten elektrischer Energie gespart, was weder erwünscht, noch sinnvoll ist.

Um diese Aufgabe bewältigen zu können, sind mehrere Schritte notwendig. Zunächst wird in Kapitel 2 ein geeignetes Modell für das Fahrzeug vorgestellt. Dies gilt als Basis für die dynamische Programmierung, die für diese Arbeit gewählte Optimierungsmethode, die in Kapitel 3 beschrieben wird. Da diese Optimierungsmethode lediglich eine optimale Steuerung für den Betrieb des Fahrzeugs liefert, die für jegliche externe Störungen eine große Anfälligkeit besitzt, wird in Kapitel 4 die modellprädiktive Regelung diskutiert. Dieser Regler integriert die Optimierungsmethode der dynamischen Programmierung in eine Reglerstruktur, die in Abhängigkeit einer im Voraus bekannten Strecke einen optimalen Betrieb des Range Extenders unter Berücksichtigung externer Störungen ermöglicht.

Kapitel 5 befasst sich mit der Vorstellung der erhaltenen Ergebnisse und mit der Überprüfung des Reglers auf Robustheit bei Parametervariationen des Modells und Änderungen des Streckenprofils.

Schließlich werden in Kapitel 6 die Erkenntnisse dieser Arbeit zusammengefasst und ein Ausblick auf zukünftige Weiterentwicklungen und Anwendungsmöglichkeiten gegeben.

Kapitel 2

Das Fahrzeugmodell

In diesem Kapitel der Arbeit wird die Idee der *quasistatischen* Modellbildung (Abschnitt 2.1), die die Grundlage für das Fahrzeugmodell bietet, erklärt. Im Abschnitt 2.2 des Kapitels werden die zugrundeliegenden Gleichungen und deren Implementierung in MATLAB® vorgestellt. Die implementierten Komponenten werden anschließend im Abschnitt 2.3 mit jenen aus der QSS-Toolbox gegenübergestellt. Die Betrachtung von Fahrstrecken für das Fahrzeugmodell wird im letzten Abschnitt 2.4 behandelt.

2.1 Quasistatische Modellbildung

Wie bereits in der Einleitung vorweggenommen, bietet sich für die gestellte Aufgabe die quasistatische Modellbildung an. Diese Art der Modellierung wird unter anderem auch in der QSS Toolbox von Guzzella und Amstutz (2005) an der ETH Zürich erfolgreich für die Modellierung von Fahrzeugen angewandt und wurde von Guzzella und Sciaretta (2005) beschrieben.

Gewöhnlich werden mathematische Modelle für dynamische Systeme mit Hilfe geeigneter Differentialgleichungen beschrieben. Betrachtet man zum Beispiel die translatorische Bewegung einer Masse m unter dem Einfluss einer Kraft F, so gilt:

$$\frac{\mathrm{d}v}{\mathrm{d}t} = \frac{F}{m} \tag{2.1}$$

Dabei betrachtet man die auf das System wirkende Kraft als Ursache für die Bewegung. Aus dieser Kraft wird die Beschleunigung bestimmt und die resultierende Geschwindigkeit v mittels Integration errechnet. Die Ursache ist also die Kraft, die Wirkung die Geschwindigkeit.

Beim quasistatischen Ansatz wird diese Wirkungsrichtung umgekehrt. Als Ursache wird die Durchschnittsgeschwindigkeit während eines Zeitschritts betrachtet, woraus die Beschleunigung und letztendlich die Kraft als Wirkung einfach wie in den Gleichungen 2.2 bestimmt werden kann. Die verschiedenen Betrachtungsmöglichkeiten der beiden Modellierungsarten ist in Tabelle 2.1 zusammengefasst.

$$a = \frac{v\left(t_2\right) - v\left(t_1\right)}{t_2 - t_1} \tag{2.2}$$

$$F = \frac{m}{a}$$

	Differentialgleichung	**Quasistatischer Ansatz**
Ursache	Kraft F (Gl. 2.1)	Durchschnittsgeschwindigkeit v (Gl. 2.2)
Wirkung	Geschwindigkeit v	Kraft F

Tabelle 2.1: Ursache und Wirkung in den verschiedenen Modellansätzen

Unter der Voraussetzung, dass der Zeitschritt $\Delta t = t_2 - t_1$ fein genug gewählt ist, lässt sich also am Beispiel eines Fahrzeugs die notwendige Antriebskraft zu jedem Zeitpunkt aus der Geschwindigkeitsanforderung und der Steigung der Fahrstrecke ermitteln. Verfolgt man diesen Ansatz im gesamten Fahrzeug für jede Komponente, erhält man ein Modell wie in Abbildung 2.1 dargestellt, wobei jeder Block in der Abbildung eine Komponente des Modells darstellt. Das Modell besteht demnach aus einem Block, der die Fahrstrecke abbildet (hellgrün), einem Block für die Betrachtung der translatorischen Bewegung des Fahrzeugs (gelb), zwei Getrieben (dunkelgrün), einem elektrischen Motor und einem Generator (rot), sowie einer Verbrennungskraftmaschine (rot) und einer Batterie (grau). Entlang der Markierungen von links nach rechts werden alle Größen nach dem quasistatischen Prinzip berechnet. Einzig die Bestimmung der Batterieladung erfolgt mit Hilfe einer Differentialgleichung (in der Abbildung mit rotem Rand hervorgehoben). Dabei wird ersichtlich, dass die Ursache in diesem Modell das Geschwindigkeitsprofil der Fahrstrecke ist und ihre Wirkung letztendlich die Änderung der Ladung in der Batterie beziehungsweise ein Kraftstoffverbrauch ist.

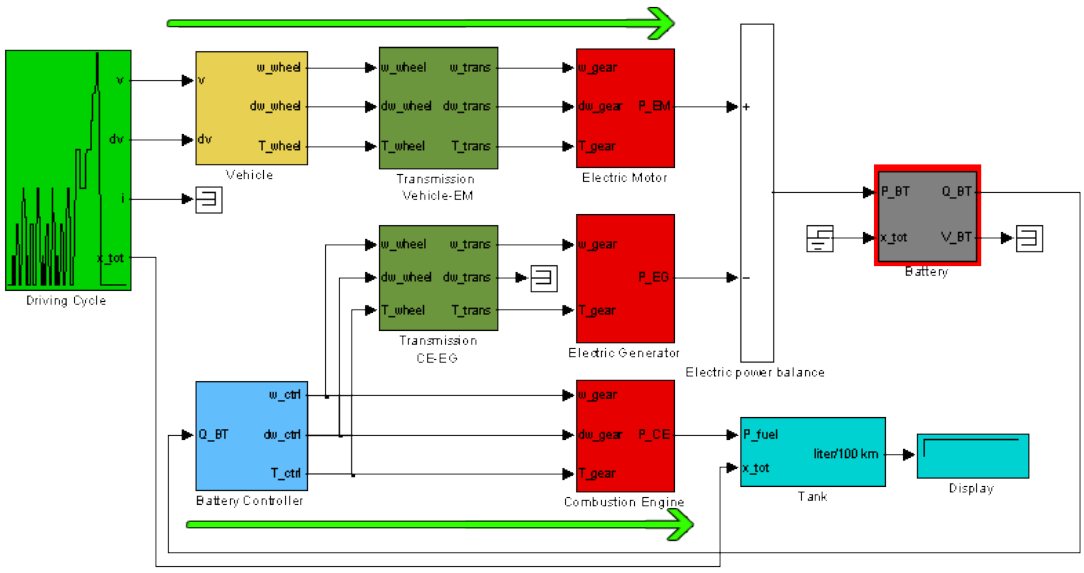
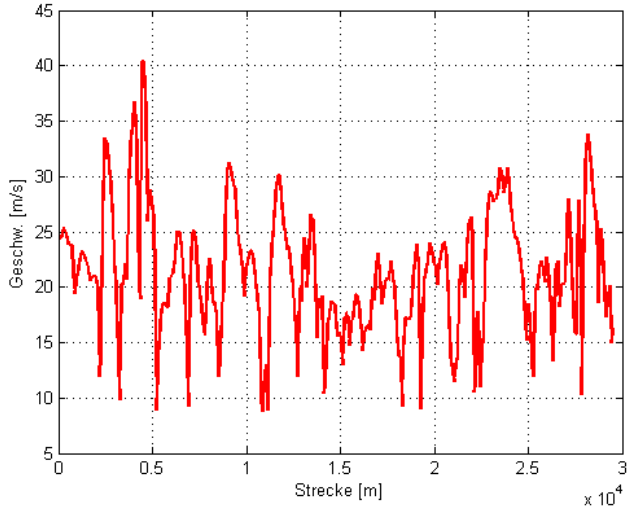


Abbildung 2.1: Fahrzeugmodell mit quasistatischer Modellierung

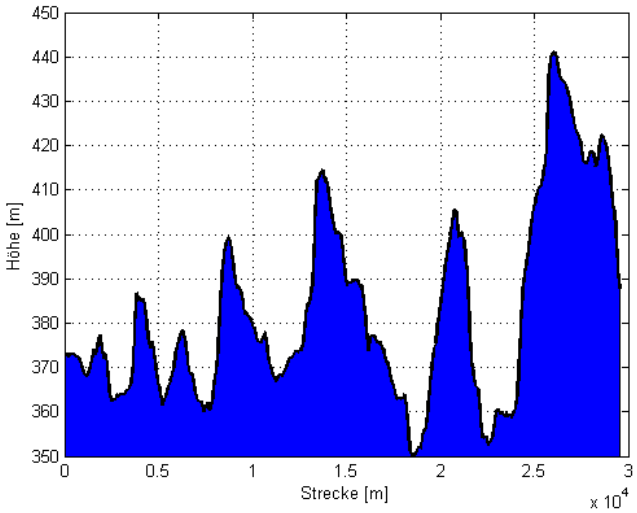
Für die weitere Vorgehensweise bei der Optimierung in Kapitel 3.3 ist es von Vorteil, das Modell nicht zeitabhängig, sondern streckenabhängig zu betrachten. Diese Idee wurde von Back, Simons, Kirschbaum, und Krebs (2002) und Back (2005) vorgestellt und angewandt. Dabei wird die geforderte Geschwindigkeit und das Höhenprofil der Strecke wie in Abbildung 2.2 als Eingangsgröße im Modell betrachtet. Zeitabhängige Größen lassen sich mit Hilfe des Zusammenhangs 2.3 in streckenabhängige Größen umrechnen. Dabei ist zu beachten, dass kleine Geschwindigkeitswerte eine Division durch Null verursachen. Bei der Umrechnung muss eine

Geschwindigkeituntergrenze gesetzt werden, deren Unterschreitung gesondert behandelt werden muss. In dieser Arbeit wurde in solchen Fällen mit einer sehr kleinen Geschwindigkeit weitergerechnet.

$$dt = \frac{1}{v} \cdot ds \tag{2.3}$$



(a) Geschwindigkeitsprofil entlang der Fahrstrecke



(b) Höhenprofil entlang der Fahrstrecke

Abbildung 2.2: Als Eingangsgrößen des quasistatischen Modells dienen das Geschwindigkeitsprofil wie in (a) und das Höhenprofil wie in (b) dargestellt. Die MATLAB®-Skriptdateien mit den einzelnen Modellen der Komponenten sind im Anhang B zu finden.

2.2 Beschreibung der Fahrzeugkomponenten

In diesem Abschnitt werden alle für die Arbeit relevanten Komponenten des Fahrzeugs beschrieben und mit Hilfe des quasistatischen Prinzips modelliert. Die Zusammenschaltung dieser Komponenten zu einem geeigneten Gesamtfahrzeugmodell wird im Abschnitt 2.2.7 beschrieben, die Einbettung des Modells in die Optimierung erfolgt im Abschnitt 3.3.

2.2.1 Translatorische Beschreibung des Fahrzeugs

Zunächst wird die Bewegungsgleichung des Fahrzeugs aufgestellt. Entsprechend dem quasistatischen Ansatz erhält das Fahrzeug die Größen der Fahrstrecke bestehend aus Geschwindigkeitsprofil $v(s)$ und Höhenverlauf als Steigungsprofil $k(s)$ wie in Abbildung 2.2 dargestellt als Ursache für die Bewegung. Daraus resultiert die Wirkung, die dem gefordertem Drehmoment M und Winkelgeschwindigkeit ω an den Rädern enstpricht. Dabei werden Verluste durch Roll- und Luftwiderstand und der Einfluss der Steigung betrachtet. Die für die Berechnung notwendigen Fahrzeugparameter sind aus der Tabelle 2.2 zu entnehmen.

Eingangsgrößen	**Parameter**	**Ausgangsgrößen**
Geschwindigkeit v $\left[\frac{m}{s}\right]$	Radradius r $[m]$	Winkelgeschwindigkeit ω $\left[\frac{rad}{s}\right]$
Beschleunigung a $\left[\frac{m}{s^2}\right]$	Fahrzeugmasse m $[kg]$	Winkelbeschleunigung $\dot{\omega}$ $\left[\frac{rad}{s^2}\right]$
Steigung k $[rad]$	Angriffsfläche A [m^2]	Drehmoment M $[Nm]$
	Luftwiderstandskoeffizient ϑ $[-]$	
	Rollwiderstandskoeffizient μ $[-]$	
	Rotierende Masse m_r [%]	
	Erdbeschleunigung g $\left[\frac{m}{s^2}\right]$	
	Luftdichte ρ $\left[\frac{kg}{m^3}\right]$	

Tabelle 2.2: Berechnungsparameter Fahrzeugmodell

Auf das Fahrzeug wirken drei Kräfte: der Luftwiderstand (2.4), der Rollwiderstand (2.5) und die Kraft durch die Steigung im Höhenprofil (2.6).

$$F_{Luft} = \frac{1}{2} \cdot \rho \cdot A \cdot \vartheta \cdot v^2 \tag{2.4}$$

$$F_{Roll} = m \cdot g \cdot \mu \cdot v \tag{2.5}$$

$$F_{Steigung} = m \cdot g \cdot sin(k) \tag{2.6}$$

Die aus der translatorischen Bewegung resultierende Rotation der Räder lässt sich durch die Umrechnung mit dem Radradius r bestimmen:

$$\omega = \frac{v}{r} \tag{2.7}$$

$$\dot{\omega} = \frac{a}{r}$$

Das Trägheitsmoment J berechnet sich aus der Massenträgheit des Fahrzeugs und dem Trägheitsmoment der drehenden Räder, deren Masse sich in Prozent des Gesamtfahrzeuggewichts (rotierende Masse m_r) ergibt:

$$J = m \cdot \left(1 + \frac{m_r}{100}\right) \cdot r^2 \tag{2.8}$$

Aus der Summe der Kräfte F_{Luft} (2.4), F_{Roll} (2.5) und $F_{Steigung}$ (2.6) und dem Trägheitsmoment J (2.8) lässt sich das geforderte Antriebsmoment M der Räder berechnen:

$$M = (F_{Luft} + F_{Roll} + F_{Steigung}) \cdot r + J \cdot \dot{\omega} \tag{2.9}$$

2.2.2 Das Getriebe

Für ein Fahrzeug mit Serienhybridantrieb ist ein Getriebe mit nur einer Untersetzungsstufe notwendig, einerseits zwischen den Antriebsrädern und dem Elektromotor, andererseits zwischen der Verbrennungskraftmaschine und dem Generator. Eingangsgrößen sind das Drehmoment M_1 und die Winkelgeschwindigkeit ω_1, die Parameter für die Modellierung sind die Getriebeübersetzung g_r und der Wirkungsgrad η_{gr}, wie in Tabelle 2.3 beschrieben.

Eingangsgrößen	Parameter	Ausgangsgrößen
Winkelgeschwindigkeit ω_1 $\left[\frac{rad}{s}\right]$	Getriebeübersetzung g_r $[-]$	Winkelgeschwindigkeit ω_2 $\left[\frac{rad}{s}\right]$
Winkelbeschleunigung $\dot{\omega_1}$ $\left[\frac{rad}{s^2}\right]$	Getriebewirkungsgrad η_{gr} $[\%]$	Winkelbeschleunigung $\dot{\omega_2}$ $\left[\frac{rad}{s^2}\right]$
Drehmoment M_1 $[Nm]$		Drehmoment M_2 $[Nm]$

Tabelle 2.3: Berechnungsparameter Getriebe

Der Zusammenhang zwischen den beiden Seiten des Getriebes wird mit folgenden Gleichungen beschrieben:

$$\omega_2 = \omega_1 \cdot g_r \tag{2.10}$$

$$\dot{\omega_2} = \dot{\omega_1} \cdot g_r$$

$$M_2 = \eta_{gr} \cdot M_1 \cdot \frac{\omega_1}{\omega_2}$$

2.2.3 Die elektrische Maschine

Die elektrische Maschine wird aus Gründen der Einsparung von Rechenzeit mit Hilfe von Kennlinien modelliert. Dabei wird aus dem geforderten Drehmoment M_{mech} und der Winkelgeschwindigkeit ω die aufzuwendende elektrische Leistung P_{EM} berechnet. Diese Kennlinien können bei der Auslegung der elektrischen Maschine ausgehend von den Normkennlinien aus Abbildung 2.3 entsprechend skaliert werden. Diese Normkennlinien bilden das maximale Drehmoment (Abb. 2.3a) und den Wirkungsgrad (Abb. 2.3b) im gesamten Betriebsbereich ab. Durch die Verwendung eines kennlinienbasierten Modells bleibt die Anzahl der Parameter wie aus Tabelle 2.4 ersichtlich auch gering, es müssen lediglich die Parameter für die Skalierung,

also der Skalierungsfaktor k_{EM}, das Drehmomentmaximum M_{max} und das Winkelgeschwindigkeitsmaximum ω_{max}, sowie das Trägheitsmoment J der Maschine und die Leistung P_{aux} möglicher zusätzlicher Verbraucher angegeben werden. Die Normkennlinie für das Drehmoment ist mit $M_{max}(\omega_r)$ gegeben, das Wirkungsgradkennfeld ist mit $\eta(\omega_r, M_r)$ definiert, diese werden entsprechend skaliert.

Eingangsgrößen	Parameter	Ausgangsgrößen
Winkelgeschwindigkeit $\omega \left[\frac{rad}{s}\right]$ Winkelbeschleunigung $\dot{\omega} \left[\frac{rad}{s^2}\right]$ Drehmoment $M_{mech}\ [Nm]$	Skalierungsfaktor $k_{EM}\ [-]$ Trägheitsmoment $J\ \left[kg \cdot m^2\right]$ Zusatzverbraucher $P_{aux}\ [W]$ Winkelgeschwindigkeitsmaximum ω_{max} $\left[\frac{rad}{s}\right]$ Drehmomentmaximum $M_{max}(\omega_r)\ [Nm]$ Wirkungsgradkennfeld $\eta(\omega_r, M_r)\ [-]$	Elektrische Leistung $P_{EM}\ [W]$ Fehlerfeld $Inf_{EM}\ [-]$

Tabelle 2.4: Berechnungsparameter elektrische Maschine

Um die Kennlinien an eine konkrete Maschine anzupassen, muss zuerst der Drehmomentbereich ausgehend von der Normkennlinie M_{max} (Abb. 2.3a) mit dem Skalierungsfaktor k_{EM} angepasst werden:

$$M_{max} = k_{EM} \cdot M_{max,Norm}$$

Die Trägheit J der elektrischen Maschine wird mit Gleichung 2.11 berücksichtigt und der zum Betriebspunkt passende Wirkungsgradwert anhand des Kennfelds $\eta(\omega, M_{EM})$ wie in Gleichung 2.12 beschrieben, bestimmt.

$$M_{EM} = M_{mech} + J \cdot \dot{\omega} \tag{2.11}$$

$$\eta_{EM} = \eta(\omega, M_{EM}) \tag{2.12}$$

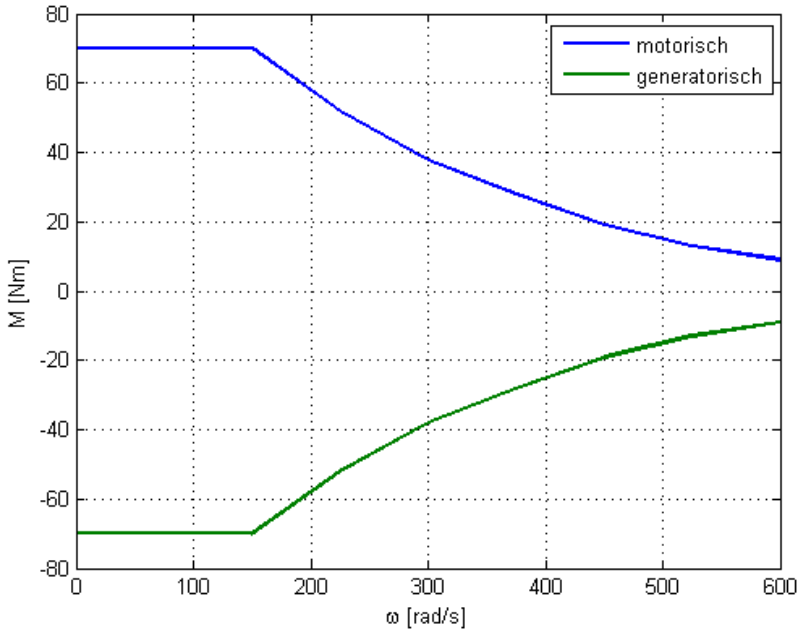
Zusätzliche elektrische Verbraucher des Fahrzeugs können mit Hilfe des Parameters P_{aux} mitmodelliert werden. Daraus folgt die notwendige elektrische Leistung P_{EM} durch die Antriebsanforderung und diese zusätzlichen Verbraucher nach:

$$P_{EM} = \eta_{EM} \cdot M_{EM} \cdot \omega + P_{aux} \tag{2.13}$$

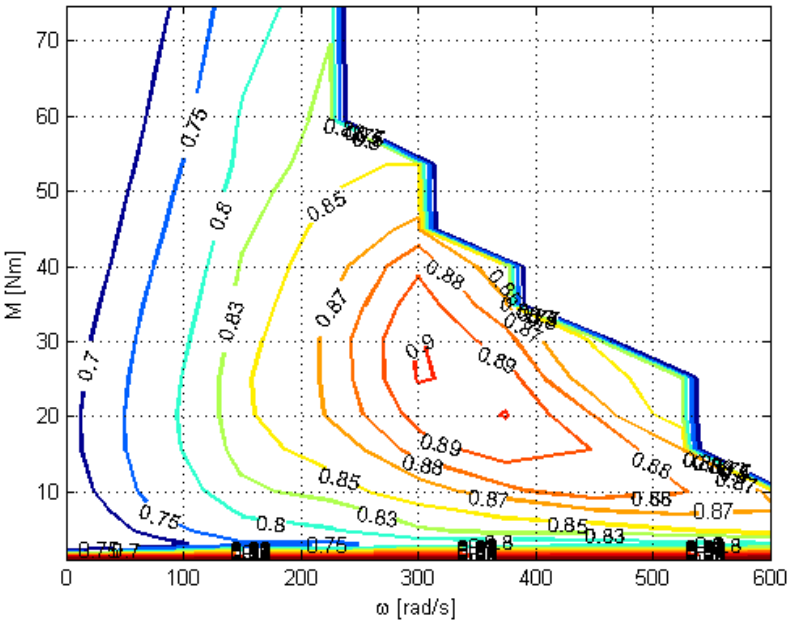
Um ungültige Betriebspunkte durch Überlastung zu erkennen und diese während der späteren Optimierung (Kapitel 3.3) berücksichtigen zu können, wird ein Fehlerfeld Inf_{EM} definiert, das bei ungültigen Betriebspunkten den Wert $Inf_{EM} = 1$ annimmt und ansonsten $Inf_{EM} = 0$ bleibt. Überlast tritt auf, wenn entweder die Winkelgeschwindigkeit ω den maximal zulässigen Wert ω_{max} überschreitet (2.14), oder das Drehmoment M_{EM} bei der aktuellen Winkelgeschwindigkeit ω den für diesen Punkt in der Drehmomentenkennlinie $M_{max}(\omega)$ definierten maximalen Wert überschreitet (2.15).

$$\omega > \omega_{max} \rightarrow Inf_{EM} = 1 \tag{2.14}$$

$$M_{EM} > M_{max}(\omega) \rightarrow Inf_{EM} = 1 \tag{2.15}$$



(a) Maximales Drehmoment im Betriebsbereich



(b) Wirkungsgrad im motorischen Betrieb

Abbildung 2.3: Kennlinien der elektrischen Maschine

2.2.4 Der elektrische Generator

Für den elektrischen Generator gelten die gleichen Zusammenhänge wie für die elektrische Maschine, mit dem Unterschied, dass der Parameter P_{aux} nicht definiert werden kann und kein motorischer Betrieb zulässig ist. Aus diesem Grund wird auf den elektrischen Generator nicht weiter eingegangen.

2.2.5 Die Verbrennungskraftmaschine

Analog zur elektrischen Maschine wird die Verbrennungskraftmaschine auch mit Hilfe von Kennlinien (Abb. 2.4) modelliert. Dabei wird eine, vom Hub V_d unabhägige Kenngröße, der mittlere effektive Druck *p* der Verbrennungskraftmaschine zum Aufspannen des Betriebsbereichs entlang der zulässigen Winkelgeschwindigkeiten ω genutzt. Dieser kann dann mit dem Hub V_d für spezifische Verbrennungskraftmaschinen skaliert werden (Gl. 2.16). Die weiteren Parameter des Modells sind der Tabelle 2.5 zu entnehmen.

Eingangsgrößen	Parameter	Ausgangsgrößen
Winkelgeschwindigkeit ω $\left[\frac{rad}{s}\right]$ Winkelbeschleunigung $\dot{\omega}$ $\left[\frac{rad}{s^2}\right]$ Drehmoment M_{mech} $[Nm]$	Skalierungsfaktor k_{ICE} $[-]$ Trägheitsmoment *J* $\left[kg \cdot m^2\right]$ Unterer Kraftstoffheizwert H_u $\left[\frac{J}{kg}\right]$ Hub V_d $\left[m^3\right]$ Leerlaufdrehzahl ω_{Leer} $\left[\frac{rad}{s}\right]$ Leerlaufleistung P_{Leer} $[W]$ Schleppmoment M_0 $[Nm]$ Leistung Schubabschaltung P_0 $[W]$ Mittlerer effektiver Druckbereich p_r $[Pa]$ Drehmomentbereich M_r $[Nm]$ Winkelgeschwindigkeitsmaximum ω_{max} $\left[\frac{rad}{s}\right]$ Mittleres effektives Druckmaximum $p_{max}\left(\omega_r\right)$ $[Pa]$ Verbrauchskennfeld $V\left(\omega_r, M_{ICE}\right)$ $\left[\frac{kg}{s}\right]$	Thermische Leistung P_{ICE} $[W]$ Fehlerfeld Inf_{ICE} $[-]$

Tabelle 2.5: Berechnungsparameter Verbrennungskraftmaschine

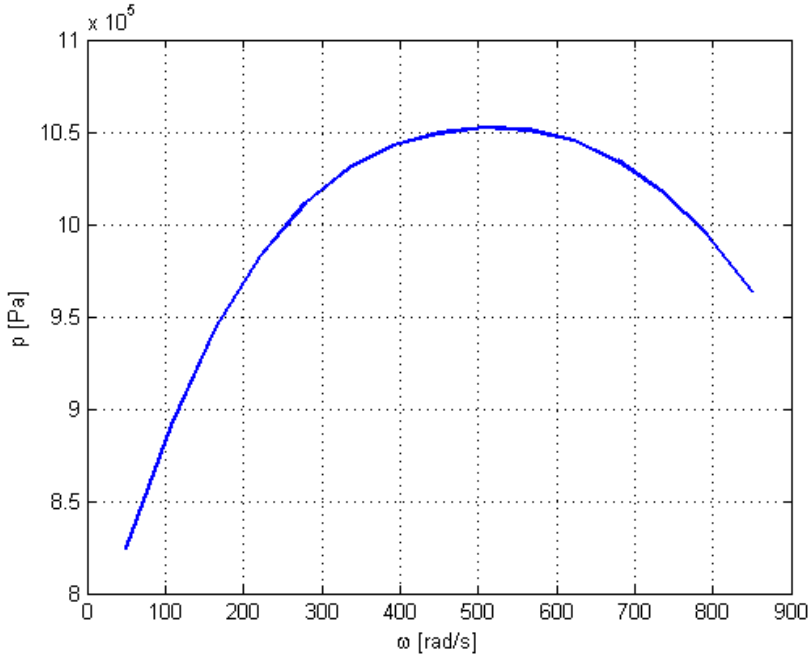
Für eine spezifische Verbrennungskraftmaschine müssen mehrere Skalierungsschritte durchgeführt werden, um die Drehmomentenkennline und die Verbrauchskennlinie zu erhalten. Dabei wird das normierte Verbrauchskennfeld V mit dem Skalierungsfaktor k_{ICE}, der Drehmomentenbereich M_r und das Drehmomentmaximum M_{max} mit dem Hub V_d an eine spezifische Maschine nach den Gleichungen 2.16 angepasst.

$$V_{map} = k_{ICE} \cdot V \tag{2.16}$$

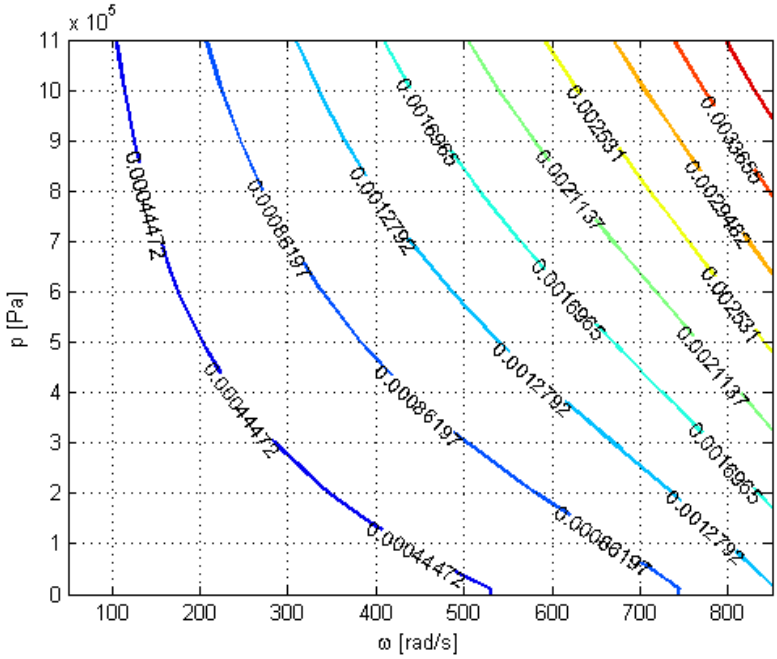
$$M_r = \frac{p_r \cdot V_d}{4\pi}$$

$$M_{max} = \frac{p_{max} \cdot V_d}{4\pi}$$

Durch das Leerlaufverhalten des Fahrzeugs ergibt sich eine untere Grenze für die Drehzahl ω_{Leer}, die von der Verbrennungskraftmaschine angefordert werden kann (Gl. 2.17). Damit ver-



(a) Maximaler mittlerer effektiver Druck im Betriebsbereich



(b) Verbrauchskennlinie im Betriebsbereich, Verbrauch in $\frac{kg}{s}$

Abbildung 2.4: Kennlinien der Verbrennungskraftmaschine

bunden kann auch das Drehmoment M_{Leer} im Leerlauf aus der Leerlaufleistung P_{Leer} und der Leerlaufdrehzahl ω_{Leer} bestimmt werden.

$$\omega < \omega_{Leer} \rightarrow \omega = \omega_{Leer} \tag{2.17}$$
$$M_{Leer} = P_{Leer} \cdot \omega_{Leer}$$

Das Gesamtdrehmoment M_{ICE} ergibt sich nach Gleichung 2.18, wobei eine Schubabschaltung durch die Unterschreitung des Schleppmoments M_0 erkannt und der in einem solchen Fall geltende Wert des Schleppmoments M_0 übernommen werden muss.

$$M_{ICE} = M_{mech} + J \cdot \dot{\omega} \tag{2.18}$$
$$M_{ICE} < M_0 \rightarrow M_{ICE} = M_0$$

Mit dem berechneten Drehmoment M_{ICE} und der Drehzahl ω lässt sich aus der Verbrauchskennlinie V_{map} der aktuelle Kraftstoffverbrauch bestimmen:

$$V_{ICE} = V_{map}\left(\omega, M_{ICE}\right) \tag{2.19}$$

Die für die geforderte Antriebsleistung P_{mech} notwendige thermische Leistung P_{ICE} berechnet sich in Abhängigkeit des unteren Heizwerts H_u des Kraftstoffs nach Gleichung 2.20, wobei hier die Fälle Leerlauf und Schubabschaltung mitberücksichtigt werden müssen. Bei beiden Fällen muss der vorgegebene Leistungswert P_{Leer} bzw. P_0 aus der Parameterliste übernommen werden.

$$P_{ICE} = V_{ICE} \cdot H_u \tag{2.20}$$
$$\omega < \omega_{Leer} \rightarrow P_{ICE} = P_{Leer}$$
$$M_{ICE} < M_0 \rightarrow P_{ICE} = P_0$$

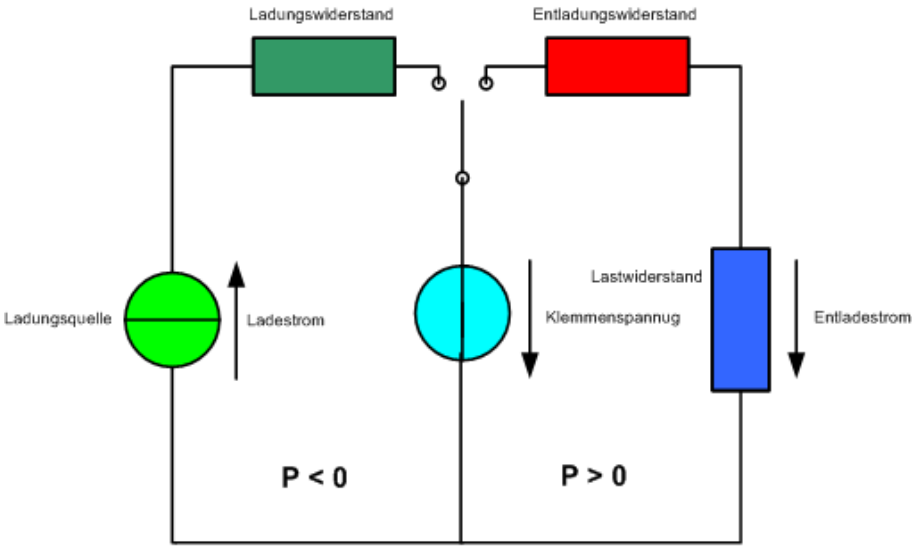
Einträge im Fehlerfeld Inf_{ICE} müssen gemacht werden, wenn die Vorgaben betreffend der maximalen Drehzahl ω_{max} oder des maximalen Drehmoments M_{max} überschritten werden (Gl. 2.21).

$$\omega > \omega_{max} \rightarrow Inf_{ICE} = 1 \tag{2.21}$$
$$M_{ICE} > M_{max} \rightarrow Inf_{ICE} = 1$$

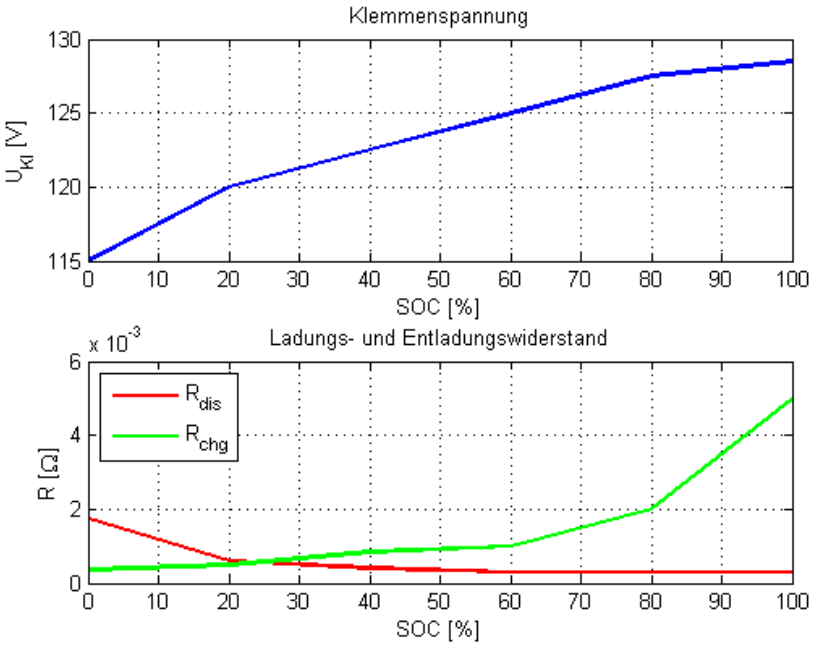
2.2.6 Die Batterie

Wie auch die anderen Fahrzeugkomponenten, wird die Ladung SOC der Batterie (Abb. 2.5a) auch mittels Kennlinien (Abb. 2.5b) ausgehend von der angeforderten elektrischen Leistung P_{BT} modelliert. Dabei wird zwischen den Betriebszuständen *Laden* und *Entladen* unterschieden, diese werden auch mittels getrennter Kennlinien behandelt. Die Parameter der Batterie sind in Tabelle 2.6 aufgelistet und beinhalten die Kennlinien des Innenwiderstands R_{BT} und den Wirkungsgrad η für beide Batteriezustände und die Batterieklemmenspannung U_{Kl} entlang des Ladungsbereichs SOC_r, sowie die Maximalladung Q_{max} und den Maximalstrom I_{max}.

Im ersten Schritt wird anhand der von der Batterie geforderten Leistung P_{BT} der Zustand der Batterie (*Laden* oder *Entladen*) bestimmt und die Parameter für den Wirkungsgrad η_{BT},



(a) Ersatzschaltbild der Batterie



(b) Kennlinien der Batterie

Abbildung 2.5: Abbildung 2.5a zeigt das Ersatzschaltbild der Batterie in den zwei Betriebszuständen. In Abbildung 2.5b werden die Klemmenspannung U_{Kl} und die Lade- und Entladewiderstände R_{chg} und R_{dis} in Abhängigkeit des Ladungszustands SOC_r dargestellt.

Eingangsgrößen	Parameter	Ausgangsgrößen
Initialladungszustand SOC_{in} [%] Batterieleistung P_{BT} [W] Zeitschritt T_s [s]	Ladungszustandspunkte SOC_r [$-$] Klemmenspannung U_{Kl} [V] Entladungskennlinie R_{dis} [Ω] Ladungskennlinie R_{chg} [Ω] Maximalladung Q_{max} [Ah] Maximalentladestrom $I_{max,dis}$ [A] Maximalladestrom $I_{max,chg}$ [A] Wirkungsgrad Entladung η_{dis} [%] Wirkungsgrad Ladung η_{chg} [%]	Ladungszustand SOC [%] Fehlerfeld Inf_{BT} [$-$]

Tabelle 2.6: Berechnungsparameter Batterie

Innenwiderstand R_{BT} und Maximalstrom I_{BT} für diesen Zustand entsprechend Tabelle 2.7 gesetzt. Dabei wurde die Vereinbarung getroffen, dass positive Leistungswerte eine Entladung, negative eine Ladung der Batterie bewirken.

$P_{BT} > 0 - Entladen$	$P_{BT} < 0 - Laden$
$\eta_{BT} = \eta_{dis}$ $R_{BT} = R_{dis}\,(SOC_{in})$ $I_{max} = I_{max,dis}$	$\eta_{BT} = \eta_{chg}$ $R_{BT} = R_{chg}\,(SOC_{in})$ $I_{max} = I_{max,chg}$

Tabelle 2.7: Batterieparameter in Abhängigkeit vom Zustand

Die Batteriespannung U_{BT} ergibt sich aus der Kennlinie der Klemmenspannung U_{Kl} beim aktuellen Ladungszustand SOC nach folgender Gleichung:

$$U_{BT} = U_{Kl}\,(SOC_{in}) \tag{2.22}$$

Der Batteriestrom I_{BT} ergibt sich aus der Batteriespannung U_{BT}, der geforderten Leistung P_{BT} und dem Verlust am Innenwiderstand R_{BT} der Batterie im aktuellen Ladungszustand wie in Gleichung 2.23 dargestellt. Da das Ergebnis der Gleichung 2.23 auch komplexe Werte annehmen kann, für die Berechnung aber nur der Realteil von Interesse ist, wird der Batteriestrom I_{BT} mit Hilfe von Gleichung 2.24 von einem etwaigen imaginären Anteil befreit, wobei mit $\bar{I}^*_{BT}$ der konjugiert komplexe Wert des Batteriestroms $\bar{I}_{BT}$ bezeichnet wird.

$$\bar{I}_{BT} = \eta_{BT} \cdot \frac{U_{BT} - \sqrt{U_{BT}^2 - 4R_{BT} \cdot P_{BT}}}{2R_{BT}} \tag{2.23}$$

$$I_{BT} = \frac{\bar{I}_{BT} + \bar{I}^*_{BT}}{2} \tag{2.24}$$

Die Änderung der Batterieladung im aktuellen Zeitschritt T_s ergibt sich aus der durch den Batteriestrom I_{BT} bewirkten Ladungsänderung. Aus der Anfangsladung SOC_{in} und der Ladungsänderung lässt sich die aktuelle Ladung SOC bezogen auf die maximale Ladung Q_{max} berechnen (Gl. 2.25).

$$SOC = -\frac{I_{BT}}{\frac{3600 \cdot Q_{max}}{T_s}} + SOC_{in} \tag{2.25}$$

Dieser Zusammenhang entspricht demnach einer diskretisierten Differentialgleichung für die Bestimmung des aktuellen Ladungszustands als dynamische Größe im Gesamtfahrzeugmodell.

Überlastzustände der Batterie können auftreten, falls der Maximalstrom I_{max} überschritten wird, oder falls die geforderte Leistung P_{BT} einen Spannungsabfall verursachen würde, der die Batteriespannung U_{BT} überschreitet (Gl. 2.26). In diesen Fällen muss ein Eintrag im Fehlerfeld Inf_{BT} gemacht werden.

$$I_{BT} > I_{max} \rightarrow Inf_{BT} = 1 \tag{2.26}$$

$$U_{BT}^2 < 4R_{BT} \cdot P_{BT} \rightarrow Inf_{BT} = 1$$

2.2.7 Zusammenfassung des Gesamtfahrzeugmodells

Aus den Komponenten der vorigen Abschnitte lässt sich nun ein Gesamtmodell eines Serienhybridfahrzeugs erstellen. Dieses Modell lässt sich prinzipiell in zwei Teilmodelle teilen: dem Antriebsstrang, bestehend aus der translatorischen Beschreibung des Fahrzeugs, dem Antriebsgetriebe und der elektrischen Maschine und dem Range Extender, bestehend aus der Verbrennungskraftmaschine und dem elektrischen Generator. Die Verbindung dieser beiden Teilmodelle stellt die Batterie über die elektrische Leistungsbilanz der elektrischen Maschine und des Generators dar.

Im Teilmodell des Antriebsstrangs sind die Eingangsgrößen als ein Geschwindigkeitsverlauf $v(s)$ und einem Steigungsprofil $k(s)$ gegeben. Diese werden mit Hilfe der translatorischen Beschreibung des Fahrzeugs aus Abschnitt 2.2.1 zur Berechnung des Antriebmoments M und der Radwinkelgeschwindigkeit ω nach folgenden Gleichungen verwendet:

$$\omega = \frac{v}{r}$$
$$M = (F_{Luft} + F_{Roll} + F_{Steigung}) \cdot r + J \cdot \dot{\omega}$$

Dabei geht das Steigungsprofil $k(s)$ in die Kraft $F_{Steigung}$ (2.6), das Geschwindigkeitsprofil in die Kräfte des Rollwiderstands F_{Roll} (2.5) und des Luftwiderstands F_{Luft} (2.4) ein.

Das Getriebe setzt nun das Antriebsmoment M und die Radwinkelgeschwindigkeit ω proportional zur Getriebeübersetzung g_r und mit Berücksichtigung des nicht idealen Wirkungsgrads η_{gr} nach den Zusammenhängen aus Abschnitt 2.2.2 in eine Drehmomentanfoderung M_{EM} bei einer Maschinenwinkelgeschwindigkeit ω_{EM} für die elektrische Antriebsmaschine um.

Die aufgenommene elektrische Leistung P_{EM} der elektrischen Maschine ergibt sich nach Bestimmung des Wirkungsgrads η_{EM} im aktuellen Betriebsbereich aus dem Wirkungsgradkennfeld $\eta_{EM}\,(\omega_{EM}, M_{EM})$ und der Berücksichtigung der Leistung P_{aux} zusätzlicher Verbraucher nach:

$$P_{EM} = \eta_{EM} \cdot M_{EM} \cdot \omega_{EM} + P_{aux} \tag{2.27}$$

Damit ist das Modell von der Antriebsseite her vollständig beschrieben.

Im Teilmodell des Range Extenders ist die Eingangsgröße als eine Drehmomentanforderung M_{ICE} bei einer Winkelgeschwindigkeit ω_{ICE} gegeben. Diese Anforderung beschreibt einen Betriebspunkt der Verbrennungskraftmaschine, in dem diese eine hohe Leistung bei hoher Effizienz zum Antrieb des elektrischen Generators aufweist. Ausgangsgröße der Verbrennungskraftmaschine ist die aufgenommene thermische Leistung P_{ICE}, die direkt mit dem Kraftstoffverbrauch proportional ist.

Aus dem Verbrauchskennfeld der Verbrennungskraftmaschine lässt sich der Verbrauch in dem durch das Drehmoment M_{ICE} und der Winkelgeschwindigkeit ω_{ICE} bestimmten Betriebspunkt, wie im Abschnitt 2.2.5 beschrieben, bestimmen

$$V_{ICE} = V_{map}\left(\omega_{ICE}, M_{ICE}\right),$$

wodurch sich die thermische Leistung P_{ICE} mit Hilfe des Kraftstoffheizwerts H_u berechnen lässt.

$$P_{ICE} = V_{ICE} \cdot H_u$$

Der elektrische Generator erhält seine Drehmomentanforderung M_{EG} und die Winkelgeschwindigkeit ω_{EG} über ein Getriebe umgesetzt aus den Anfordeungen an die Verbrennungskraftmaschine (M_{ICE} und ω_{ICE}). Die Leistung des Generators P_{EG} wird analog zu der Leistung der elektrischen Maschine nach Gleichung 2.27 berechnet, womit auch der Range Extender vollständig beschrieben ist.

Die elektrische Leistungsbilanz dient als Eingangsgröße der Batterie und bestimmt die Leistung P_{BT} dieser nach $P_{BT} = P_{EM} + P_{EG}$. Anhand dieser Leistung P_{BT} und dem aktuellen Ladungszustand SOC der Batterie lässt sich schließlich die Änderung der Batterieladung nach Abschnitt 2.2.6 berechnen, wodurch die beiden Teilmodelle zusammengeführt worden sind und die Zustandsgröße SOC bestimmt ist.

2.3 Evaluierung des Modells anhand der QSS-Toolbox

Die Modelle der Komponenten aus Kapitel 2.2 wurden als MATLAB®-Skripts implementiert. Das bedeutet, dass alle Eingangsvariablen nicht nur skalare, sondern auch vektorielle Größen annehmen können, damit sind gleichzeitige Berechnungen am gesamtem Zustandsgitter (bzw. an mehreren Punkten des Zustandsgitters) möglich. Das ist ein wichtiger Vorteil gegenüber Simulink-basierten Modellen und wird für die Optimierung in Kapitel 3.3 eine wichtige Rolle spielen.

Um die Funktion dieser MATLAB®-Skripts zu evaluieren, werden in diesem Abschnitt Vergleiche mit Ergebnissen aus der QSS-Toolbox von Guzzella und Amstutz (2005) dargestellt. Da die QSS-Toolbox das Fahrzeug mit Hilfe des quasistatischen Ansatzes (Abschnitt 2.1) unter Simulink modelliert, kann man eine gute Übereinstimmung erwarten, wobei durch die unterschiedlichen Rechenmethoden leichte Abweichungen möglich sind.

Für die Evaluierung wurde ein einfaches Fahrprofil, wie in Abbildung 2.7a dargestellt, gewählt. Da die QSS-Toolbox kein Höhenprofil verarbeiten kann, wurden lediglich zwei Beschleunigungs- und Bremsvorgänge vorgegeben. Diese Bremsvorgänge sind ausreichend, um die elektrische Maschine auch in den generatorischen Betriebsbereich zu bringen, wodurch die Batterie geladen wird. Damit sind bei allen Komponenten alle Betriebsmodi abgedeckt.

Evaluiert wurden die Komponenten elektrische Maschine, Verbrennungskraftmaschine und Batterie, wobei das Getriebe und das Fahrzeugmodell bei allen drei Komponenten ausgehend vom Fahrprofil in Abbildung 2.7a mitberechnet worden sind. Die QSS-Modelle für diese Komponenten sind in Abbildung 2.6 dargestellt.

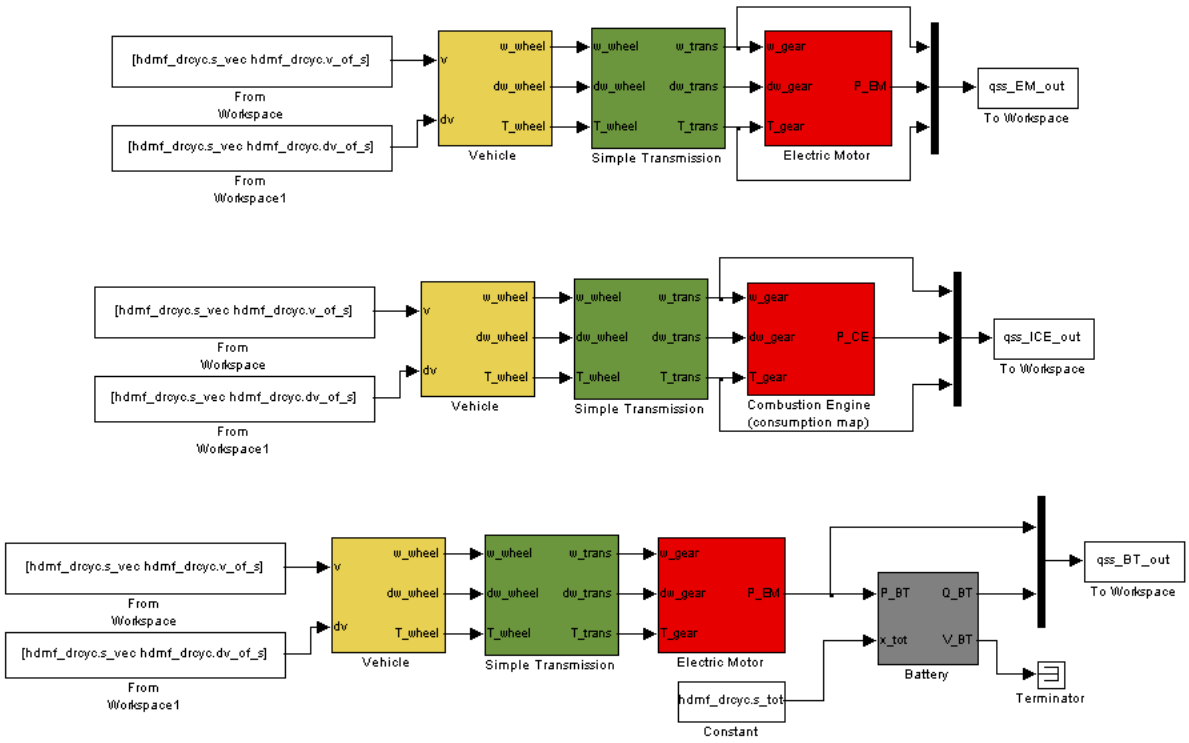


Abbildung 2.6: QSS-Modelle der Komponenten elektrische Maschine, Verbrennungskraftmaschine und Batterie

In der Abbildung 2.7b-c sind die Verläufe der elektrischen Maschine und der Verbrennungskraftmaschine aus der QSS-Toolbox mit den Ergebnissen des MATLAB®-Skripts gegenübergestellt. Abbildung 2.7d zeigt den Vergleich des Batteriemodells.

Wie erwartet, entsprechen die Ergebnisse aus dem MATLAB®-Skript denen aus der QSS-Toolbox. Lediglich bei den Verläufen der elektrischen Maschine und der Verbrennungskraftmaschine ist im stationären Bereich eine konstante Abweichung festzustellen. Diese Abweichung kann auf die verschiedenen Rechenmethoden zurückgeführt werden und ist für die weitere Verwendung des Modells unkritisch, da im Verlauf der Batterieladung, die die auschlaggebende Größe während der Optimierungsaufgabe sein wird, diese Abweichung keinen nennenswerten Einfluss hat.

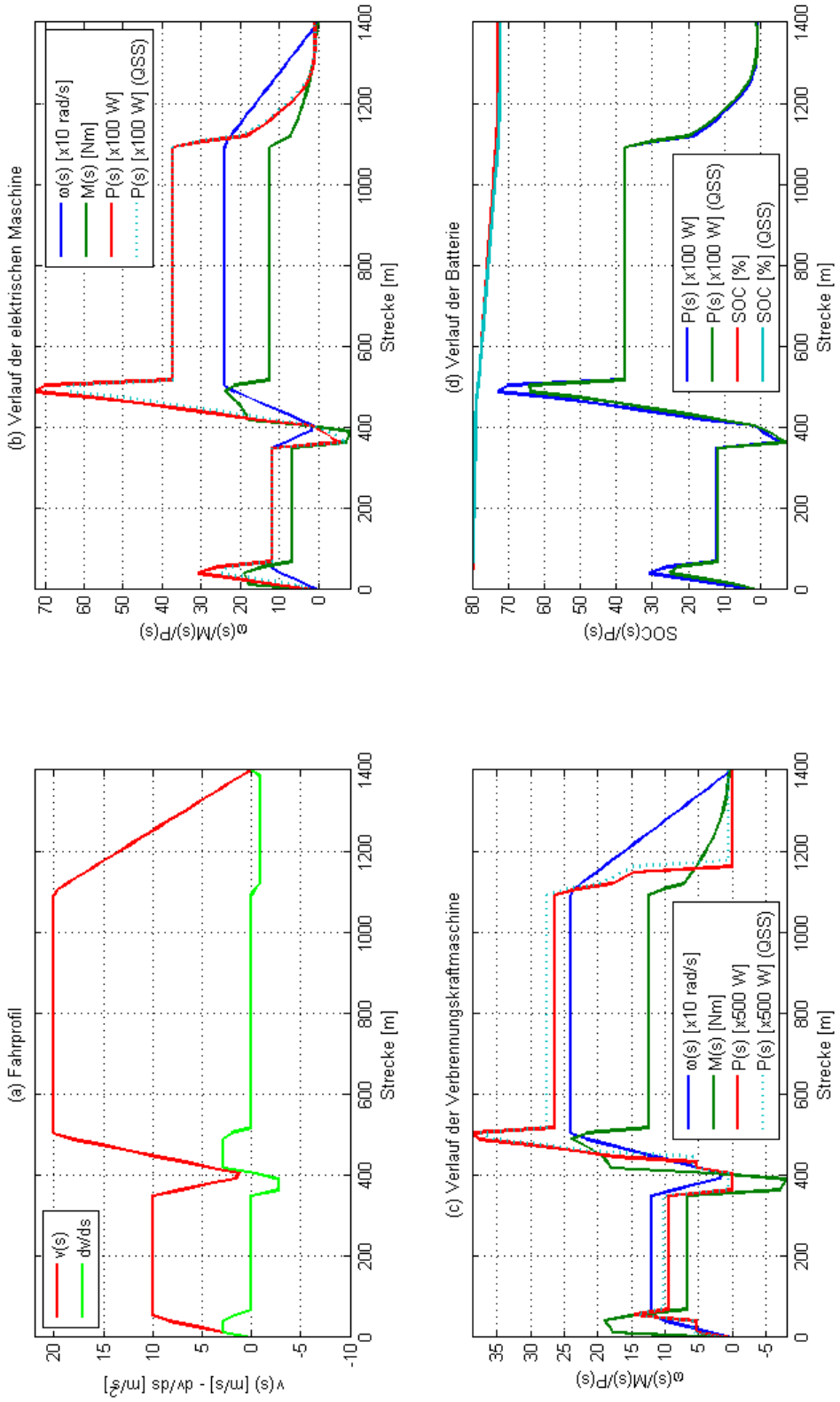


Abbildung 2.7: Vergleich der Verläufe aus der QSS-Toolbox und dem MATLAB ®-Skript

2.4 Erfassung und Einbindung von Fahrstrecken

Um den in Kapitel 4 entworfenen Regler realitätsnah testen zu können, müssen reale Fahrstrecken aufgenommen und dem Modell entsprechend in eine streckenabhängige Form nach Gleichung 2.3 gebracht werden. Diese Strecken müssen einerseits ein Geschwindigkeitsprofil andererseits auch ein Höhenprofil enthalten und einer realen Fahrstrecke entsprechen, um in Kapitel 5.1 visualisiert werden zu können.

Um die Grundfunktionen des Reglers zu testen, wurde eine kurze Teststrecke (Abb. 2.9) an der B64 - Rechberger Straße, Steiermark[1] benutzt, die alle Betriebszustände des Fahrzeugs anspricht.

Für weitere Tests und die Simulation von Umleitungen in der Fahrstrecke während der Regelung (siehe Kapitel 5) wurden zwei andere Strecken abgefahren. Die erste Strecke verläuft in Budapest, Ungarn (Abb. 2.11) und besitzt die Eigenschaften einer Stadtfahrt mit geriner Durchschnittsgeschwindigkeit (Abb. 2.12a) und einer geringen Höhenänderung, wobei während der Umleitung im Höhenprofil ein gut erkennbarer Hügel vorhanden ist (Abb. 2.12b). Die zweite Strecke verläuft von Graz nach Wolfsberg im Schwarzautal, Steiermark (Abb. 2.13) und beinhaltet eine Landstraßenfahrt mit höherer Durchschnittsgeschwindigkeit (Abb. 2.14a) und ein stark wechselndes Höhenprofil (Abb. 2.14b).

```
1       <trkpt lat="47.26769000" lon="15.33202200">
2         <ele>444.00000</ele>
3         <time>2011-05-22T15:37:19Z</time>
4       </trkpt>
```

Abbildung 2.8: Ein GPX-Streckenpunkt

Die Daten aller Fahrstrecken liegen im GPX-Format vor, dessen genaue Struktur von Foster (2011) beschrieben worden ist. Das GPX-Format ist ein einfaches, lizenzfreies XML-Format für die textuelle Speicherung, die Verarbeitung und den Austausch von GPS-Daten. Eine GPX-Datei enthält Streckenpunkte (Abb. 2.8), die aus geographischen Koordinaten, der Seehöhe und einem Zeitstempel bestehen. Die Speicherung der Streckenpunkte erfolgt gerätespezifisch zu fixen Zeiten und/oder nach Zurücklegen einer bestimmten Distanz. Für die Aufnahme der Streckenpunkte diente ein *Nokia 5800* Navigationshandy mit der Software *SportsTracker*.

Aus den gespeicherten Streckenpunkten lässt sich anschließend die Strecke mit Hilfe von MATLAB® rekonstruieren.[2] Da der Weg in Meilen statt Kilometern berechnet wird, muss eine Umrechnung und Offsetbefreiung der Einheiten und eine Interpolation auf eine Darstellung mit Streckenpunkten fixen Abstands stattfinden. Aus dem Höhenprofil h lässt sich mit Hilfe der Gleichung $k = arctan\left(\frac{dh}{dx}\right)\ [rad]$ ein Steigungsprofil k einfach berechnen. Nach diesen Schritten sind die Streckendaten in einer für die weitere Verarbeitung vorteilhafte Form gebracht.

[1] An dieser Stelle speziellen Dank an den GPSies.com Benutzer *Rene* für die Veröffentlichung dieser Strecke (`http://www.gpsies.com/map.do?fileId=gruseacxncnntsdc`).

[2] An dieser Stelle speziellen Dank an den MATLAB® Central Benutzer *karl critz* für die Veröffentlichung seiner Skriptdatei `loadgpx` (`http://www.mathworks.com/matlabcentral/fileexchange/24154-gpx-file-reader`).

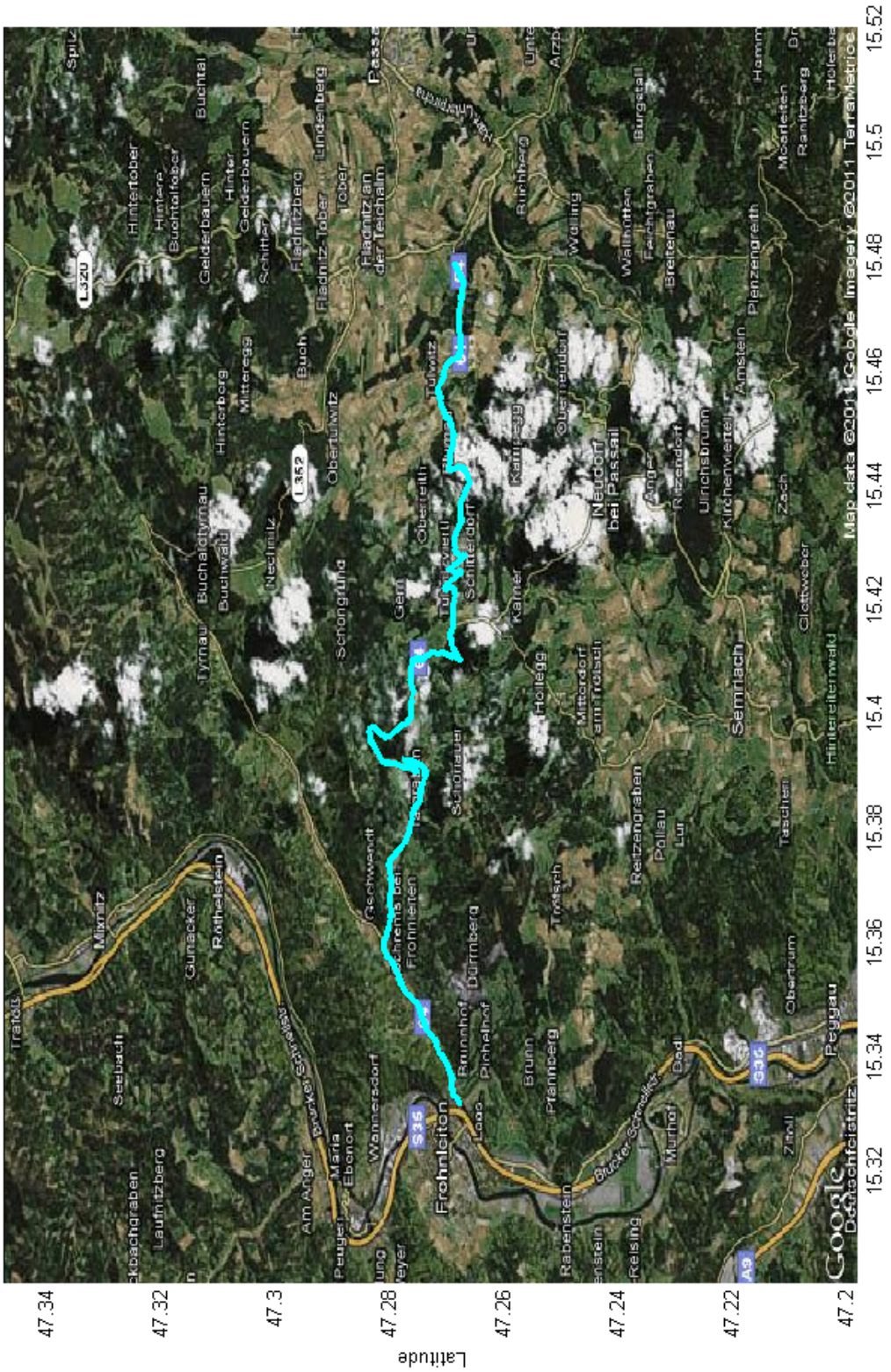
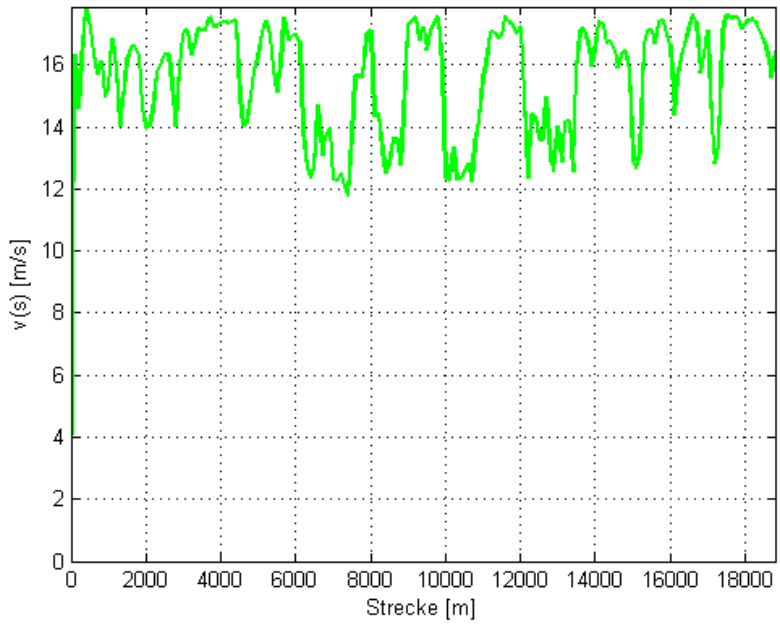
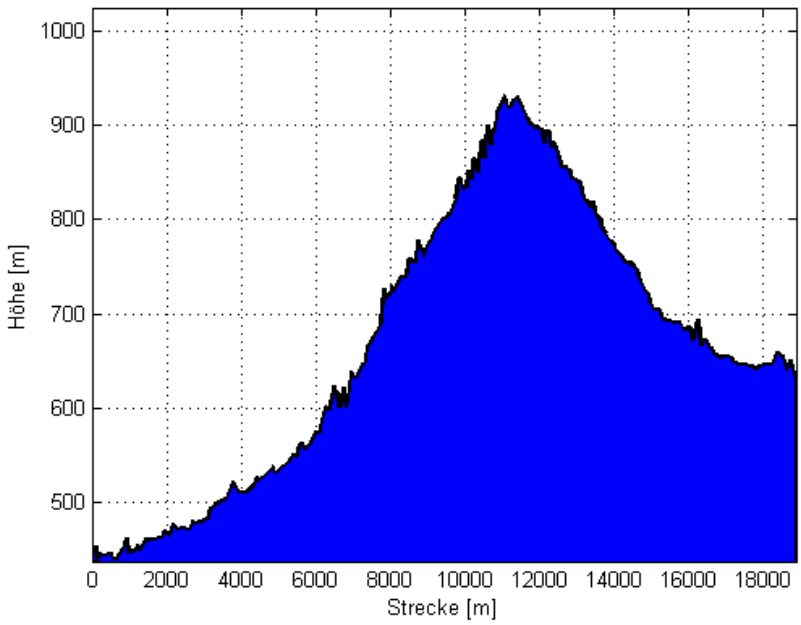


Abbildung 2.9: Verlauf der B54 - Rechberger Straße



(a) Geschwindigkeitsprofil B54 - Rechberger Straße



(b) Höhenprofil B54 - Rechberger Straße

Abbildung 2.10: Daten der Rechberg-Strecke

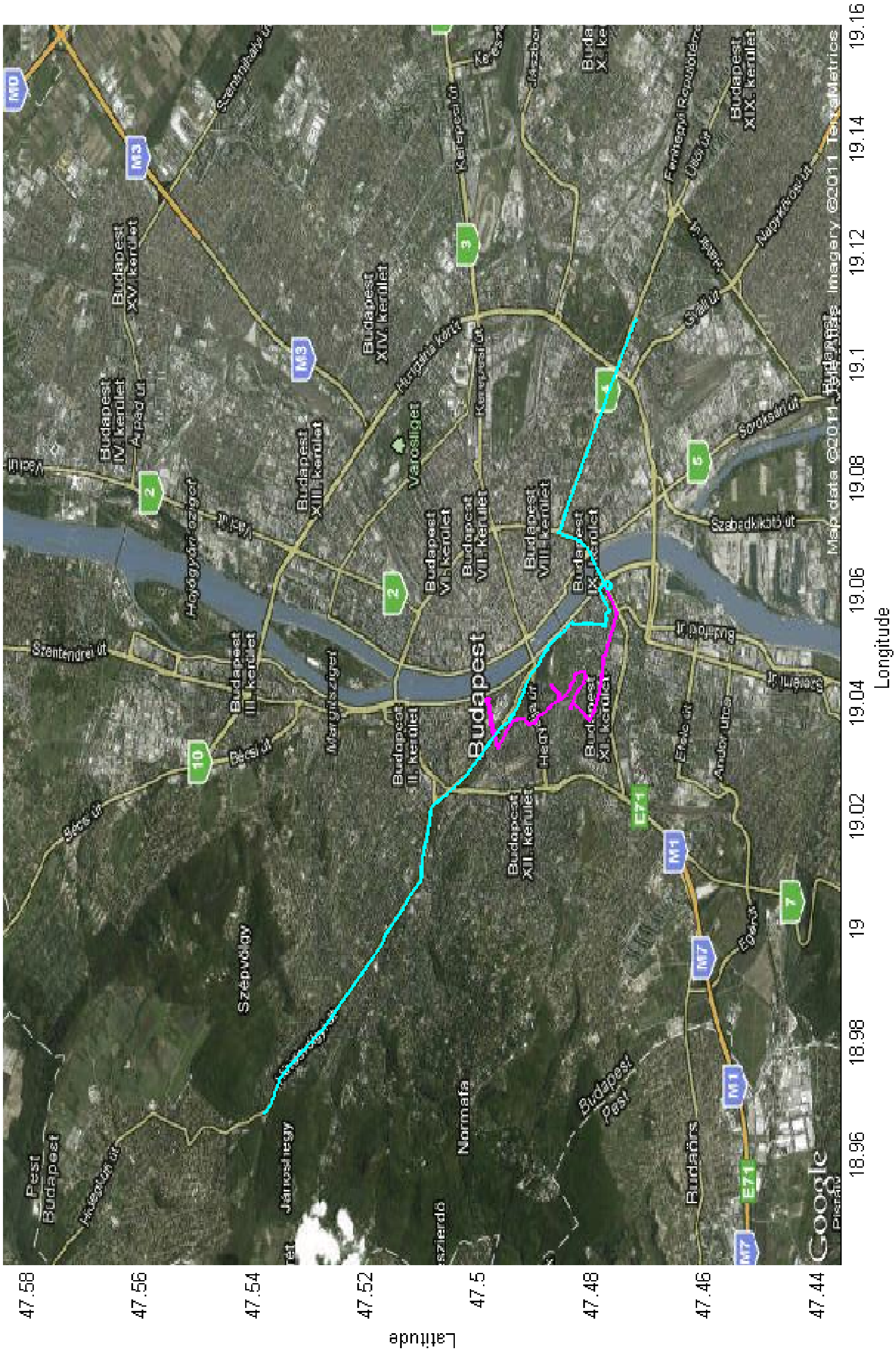
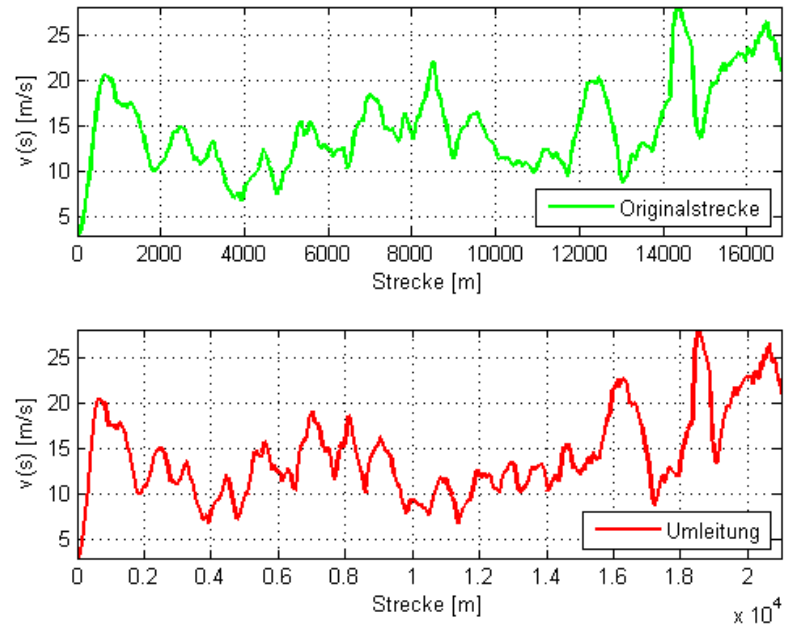
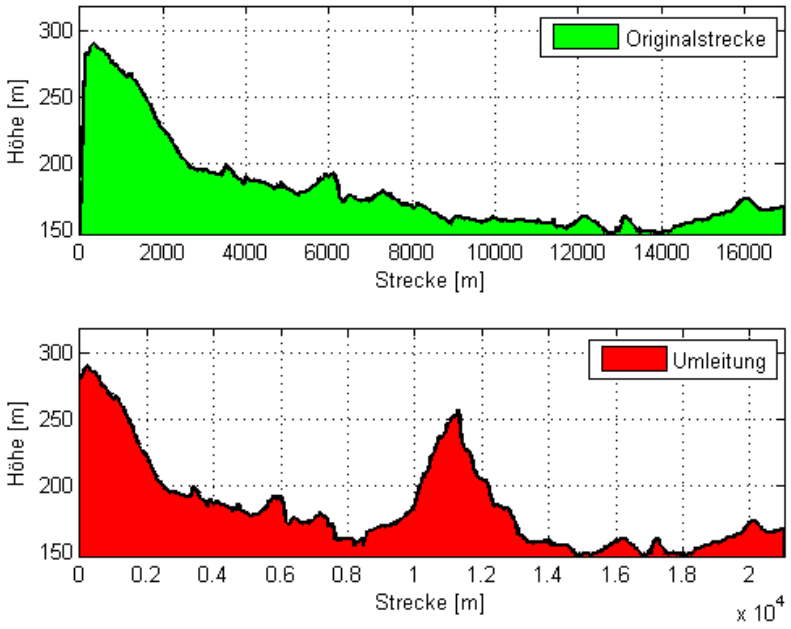


Abbildung 2.11: Verlauf der Budapest Strecke mit Umleitung



(a) Geschwindigkeitsprofil der Budapest Strecke



(b) Höhenprofil der Budapest Strecke

Abbildung 2.12: Daten der Budapest-Strecke

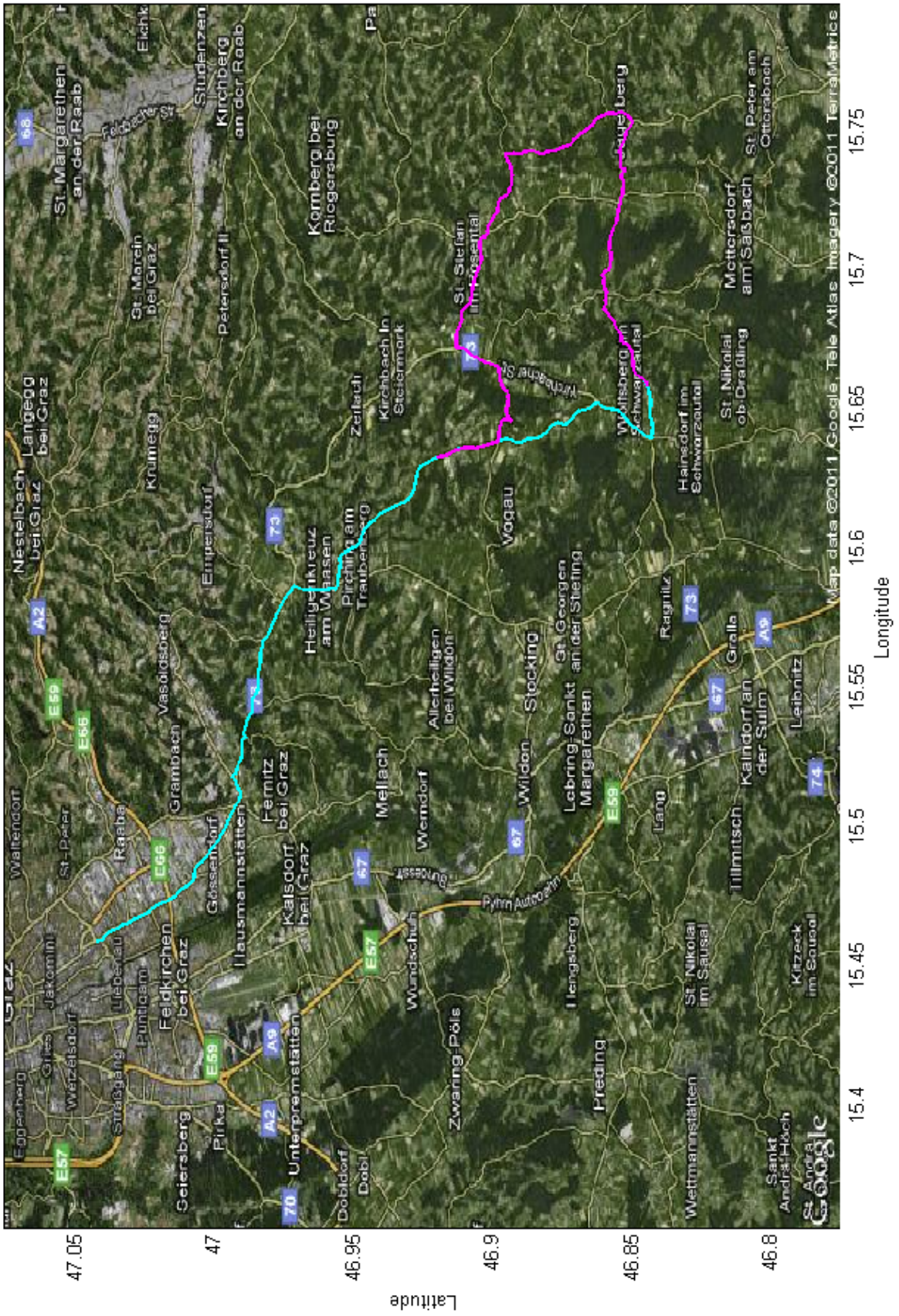
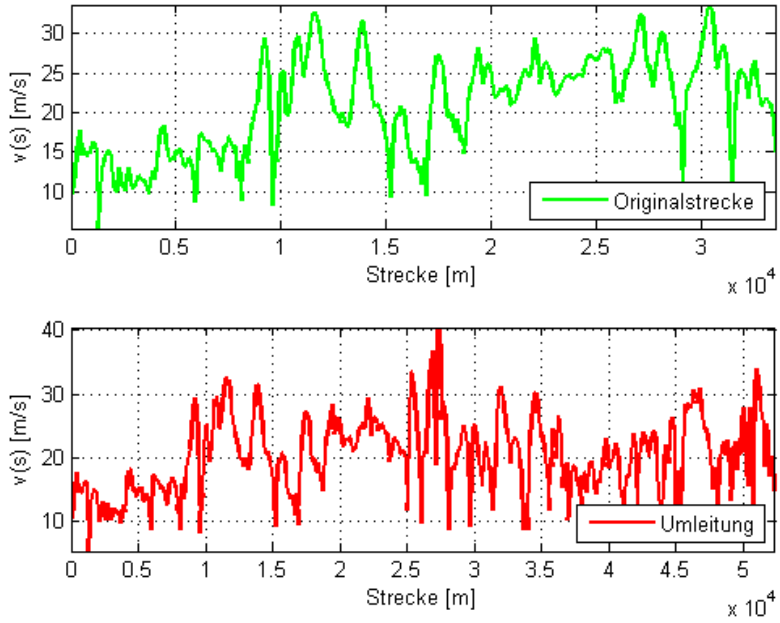
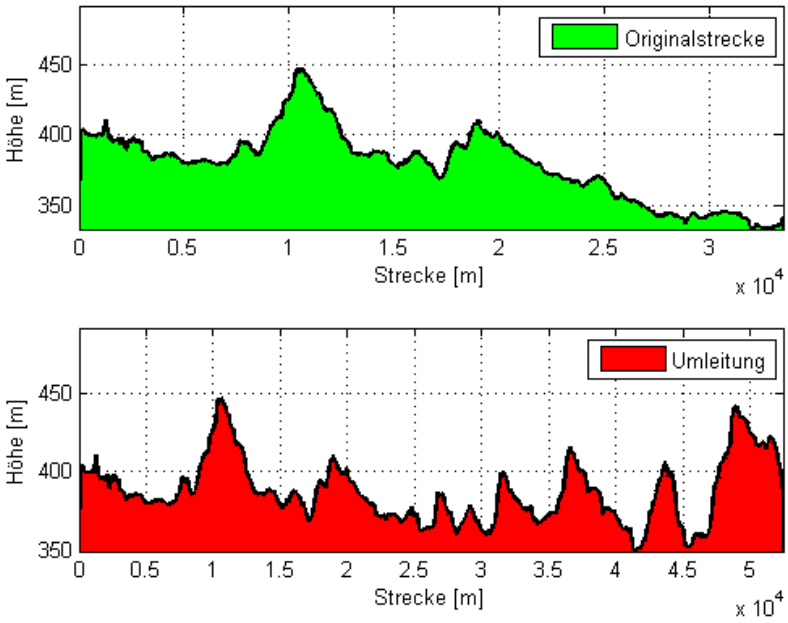


Abbildung 2.13: Verlauf der Graz-Wolfsberg Strecke mit Umleitung



(a) Geschwindigkeitsprofil der Graz-Wolfsberg Strecke



(b) Höhenprofil der Graz-Wolfsberg Strecke

Abbildung 2.14: Daten der Graz-Wolfsberg-Strecke

Kapitel 3

Dynamische Programmierung

In diesem Kapitel wird die dynamische Programmierung nach dem Optimalitätsprinzip von Bellman beschrieben. Die grundlegende Idee und die allgemeine mathematische Formulierung wird im Abschnitt 3.1 erläutert und mit Hilfe eines einfachen Beispiels illustriert. Im Abschnitt 3.2 wird die für die Lösung gewählte Optimierungsfunktion `dpm.m` in MATLAB® beschrieben. Die Anwendung der dynamischen Programmierung auf das Modell eines Serienhybridfahrzeugs wird im Abschnitt 3.3 beschrieben und die dabei erhaltenen Ergebnisse vorgestellt.

3.1 Grundlagen der dynamischen Programmierung

Die dynamische Programmierung ist auf das Optimalitätsprinzip von Bellman aus dem Jahr 1957 zurückzuführen, das folgendes besagt:

> *„Eine optimale Entscheidungsfolge hat die Eigenschaft, dass, wie auch immer der Anfangszustand war und die erste Entscheidung ausfiel, die verbleibenden Entscheidungen eine optimale Entscheidungsfolge bilden müssen, bezogen auf den Zustand, der aus der ersten Entscheidung resultiert.“ - Bellman, 1957*

Daraus folgt, dass die optimale Lösung eines komplexen Gesamtproblems durch Aufteilung und Lösung kleinerer Teilprobleme vereinfacht werden kann. Durch Zusammenfügung der Teilergebnisse erhält man anschließend die Gesamtlösung des Optimierungsproblems.

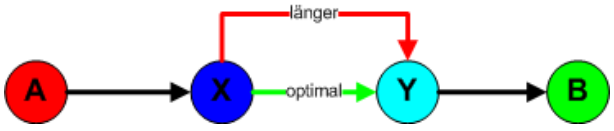


Abbildung 3.1: Einfaches Wegnetz

Dieses Prinzip lässt sich anhand der Suche nach dem kürzesten Weg in einem Wegnetz veranschaulichen. In Abbildung 3.1 ist ein einfaches Wegnetz dargestellt. Die Aufgabenstellung soll lauten: *finde den kürzesten Weg zwischen den Städten A und B*. Nehmen wir an, wir wissen, dass die Ortschaften X und Y auf dem kürzesten Weg zwischen den Städten A und B liegen. In diesem Fall reduziert sich die Aufgabe auf die Suche nach dem kürzesten Weg zwischen X und Y. Lässt sich nämlich der Weg zwischen X und Y durch eine andere Wahl des Weges

weiter verkürzen, ist dieser Teilweg automatisch Bestandteil des kürzesten Weges zwischen den Städten *A* und *B*.

Diese Vorgangsweise lässt sich auch auf ein komplexeres Wegnetz aus Abbildung 3.2 anwenden, wie es als Beispiel von Schneider und Mikolcić (1972) beschrieben worden ist. Gesucht ist der kürzeste Weg zwischen den Punkten *A* und *B*, wobei den einzelnen Wegstücken verschiedene Gewichtungen $C\,(x,k)$ zugeordnet sind. Ziel ist die Minimierung der Kostenfunktion $J\,(A,B)$, die durch die Summe der Gewichtungen $C\,(x,k)$ entlang des gewählten Weges gegeben ist:

$$J = \sum_{k=0}^{6} C\,(x,k)$$

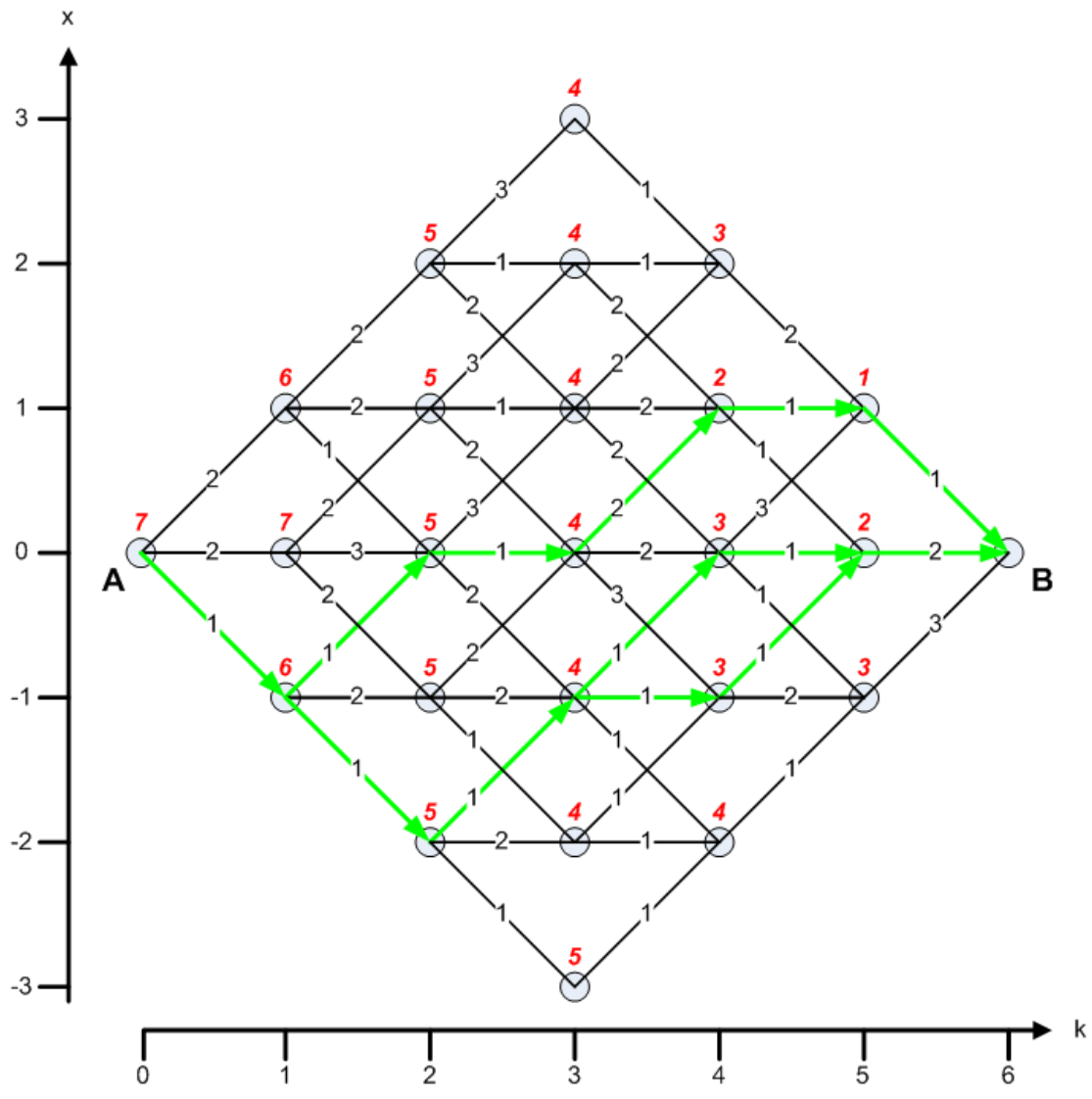


Abbildung 3.2: Wegnetz mit mehrstufigem Entscheidungsprozess

Ausgehend von Punkt *B* werden alle Punkte entlang der Abszisse k abgefahren und in jedem Punkt die minimalen Kosten $J(x,k)$ für den Weg u_k berechnet, die von diesem Punkt $P(x,k)$ zum Endpunkt *B* führen. Diesen Vorgang, wobei vom Endpunkt *B* ausgegangen wird, nennt man *Rückwärtsrechnung*. Folgende Schritte zeigen den Weg zur Bestimmung der optimalen Lösung:

Schritt 1: Ausgehend von Punkt *B* führt jeweils ein Weg in die Punkte $P(1,5)$, $P(0,5)$ und $P(-1,5)$, woraus sich die Kostenwerte für diese Punkte sofort als $J(1,5)=1$, $J(0,5)=2$ und $J(-1,5)=3$ ergeben. Diese Kostenwerte werden für die jeweiligen Punkte hinterlegt (in der Abbildung 3.2 mit rot markiert) und dienen als Wegweiser, mit deren Hilfe eine Aussage darüber getroffen werden kann, mit welchen restlichen Kosten ausgehend vom jeweiligen Punkt bis zum Punkt *B* gerechnet werden muss, wenn man den Weg über den jeweiligen Punkt einschlägt.

Schritt 2: untersuche die Wege ausgehend von den Punkten $P(x,4)$, wie im Folgenden am Beispiel zweier Punkte dargestellt:

- Ausgehend von Punkt $P(0,4)$: der kürzeste Weg aus den drei möglichen Wegen *aufwärts* zum Punkt $P(1,5)$, *gerade* zum Punkt $P(0,5)$ oder *abwärts* zum Punkt $P(-1,5)$ ist zu wählen. Addiert man die Gewichte dieser Wege zum Kostenwert $J(x,5)$ des jeweiligen Punktes $P(x,5)$ zu dem der Weg führt, kommt man zum Schluss, dass der *gerade* Weg über den Punkt $P(0,5)$ der kürzeste Weg zum Endpunkt *B* sein muss.
- Ausgehend von Punkt $P(-1,4)$: analog zum Punkt $P(0,4)$ werden die Kosten bestimmt, die mit der Wahl der Wege *gerade* oder *aufwärts* verbunden sind. Daraus folgt, dass der *aufwärts* führender Weg über den Punkt $P(0,5)$ mit geringeren Kosten verbunden ist.

Schritt 3: Analog zu *Schritt 2* werden alle Wege zwischen den verbleibenden Punkten ausgewertet und führen letztendlich bei jedem Punkt $P(x,k)$ zu einem Kostenwert $J(x,k)$, der die restlichen, *minimalen* Kosten bis zum Endpunkt *B* bei der Wahl des Weges über den jeweiligen Punkt $P(x,k)$ zeigt.

Schritt 4: Ist man am Punkt *A* angelangt, hat man alle Punkte $P(x,k)$ mit einem Wegweiser versehen, die die *minimalen* Kosten durch die Wahl der jeweiligen Richtung zeigen. Die Aufgabe besteht nurmehr darin, ausgehend von Punkt *A* jene Wege durchzulaufen, die durch die Punkte mit den niedriegsten Kostenwerte führen. Wie in Abbildung 3.2 dargestellt, ist es möglich, dass mehrere Wege mit dem gleichen minimalen Kostenwert zum Endpunkt *B* führen. In diesem Fall führen die Wege über $P(-1,1)-P(0,2)-P(0,3)-P(1,4)-P(1,5)-P(0,6)$, $P(-1,1)-P(-2,2)-P(-1,3)-P(0,4)-P(0,5)-P(0,6)$ und $P(-1,1)-P(-2,2)-P(-1,3)-P(-1,4)-P(0,5)-P(0,6)$ zum gleichen optimalen Kostenfunktionswert.

Bemerkenswert bei der Anwendung der Rückwärtsrechnung ist die Tatsache, dass falls der Anfangspunkt *A* nicht fixiert wird, diese Vorgangsweise automatisch zu jedem möglichen Anfangspunkt die minimalen Kosten zwischen dem jeweiligen Anfangspunkt und dem Endpunkt *B* bestimmt. Im Gegensatz dazu kann diese Vorgangsweise in die sogenannte Vorwärtsrechnung umgewandelt werden, wobei ausgehend von einem fixierten Anfangspunkt *A* die minimalen Kosten automatisch zu allen möglichen Endpunkten bestimmt werden. Dadurch kann man

Suche nach dem kürzesten Weg	**Mathematisches Modell**
Wegnetz	System
Wahl eines Teilweges ausgehend von einem bereits erreichten Punkt	Stellgröße u_k
Bestimmung des kürzesten Weges zum Endpunkt	N-stufiger Entscheidungsprozess
Punkte die im aktuellen Schritt erreicht werden können	Zustand x_k
Gewicht des aktuellen Teilweges	Kostenwert J_k zur aktuellen Stellgröße u_k
Summe der Gewichte entlang eines Gesamtweges	Kostenfunktion *J* entsprechend der Stellfolge $u = \{u_0, u_1, ..., u_N\}$

Tabelle 3.1: Übertragung des Wegnetzproblems auf ein mathematisches Modell

zwischen Rückwärts- und Vorwärtsrechnung entsprechend den Anforderungen der gestellten Optimierungsaufgabe wählen. In dieser Arbeit wird jedoch ausschließlich die Methode der Rückwärtsrechnung verwendet.

Betrachtet man ein mathematisches Modell der Form von Gleichung 3.1, kann man die Vorgangsweise zur Aufstellung des kürzesten Weges mit Hilfe von Tabelle 3.1 auch auf ein solches System abbilden.

$$x_{k+1} = F_k\left(x_k, u_k\right), \;\; k = 0, 1, ..., N-1 \tag{3.1}$$

$$J_k = min \sum C\left(x_k, u_k\right)$$

Die Lösung des Optimierungsproblems im mathematischen Modell erfolgt demnach nach folgender Vorschrift:

Start: Auflösung des Problems beginnend mit $k = N$ beim Endzustand x_N.

Schritt 1: Untersuche die Zustandsübergänge zwischen x_k und x_{k-1}. Bestimme den optimalen Kostenwert J_k zu jeder möglichen Stellgröße u_k aus der Menge $u = \{u_0, u_1, ..., u_N\}$.

Schritt 2: $k = k - 1$ - falls $k > 0$, gehe zu *Schritt 1*, sonst *Ende*.

Ende: Der Zustandsraum wurde durchlaufen, die verbleibenden Stellgrößen der Menge $u = \{u_0, u_1, ..., u_N\}$ sind optimale Lösungen.

3.2 Beschreibung der Funktion `dpm.m`

Für die Lösung eines Optimierungsproblems aus Abschnitt 3.1 bietet sich die MATLAB®-Funktion `dpm.m` an, die am Institut für Mess- und Regeltechnik von Sundström und Guzzella (2009) entwickelt worden ist. Diese Funktion bietet die Möglichkeit zur Einbindung von nichtlinearen, zeitinvarianten mathematischen Modellen als MATLAB®-Skripts, die in der Form von 3.1-3.3 formuliert werden können und unterstützt Modelle bis zur 5-ten Ordnung.

Analog zu Gleichung 3.1 aus Abschnitt 3.1 ist das System folgendermaßen definiert:

$$x_{k+1} = F_k\left(x_k, u_k\right), \;\; k = 0, 1, ..., N-1$$

Nehmen wir an, dass Gleichung 3.2 eine zulässige Steuerfolge für das System ist:

$$\pi = \{\mu_0, \mu_1, ..., \mu_{N-1}, \} \tag{3.2}$$

Die optimale Steuerfolge π^O ist in diesem Fall dadurch bestimmt, dass sie die Kostenfunktion

$$J_\pi(x_0) = g_N(x_N) + \phi_N(x_N) + \sum_{k=0}^{N-1} \left[h_k(x_k, \mu_k(x_k)) + \phi_k(x_k)\right] \tag{3.3}$$

minimiert:

$$J^O(x_0) = \min_{\pi \in \Pi} J_\pi(x_0) \tag{3.4}$$

Dabei wird mit dem Term $g_N(x_N) + \phi_N(x_N)$ der Endzustand bestraft. Der erste Term $g_N(x_N)$ entspricht dem Kostenwert im Endzustand, mit dem zweiten Term $\phi_N(x_N)$ kann ein zulässiger Bereich für den Endzustand erzwungen werden. Die Kosten für die Beaufschlagung mit einer spezifischen Stellgröße $\mu_k(x_k)$ werden durch den Term $h_k(x_k, \mu_k(x_k))$ quantifiziert. Zuletzt können Begrenzungen der Zustandsvariablen x_k mit dem Term $\phi_k(x_k)$ erzwungen werden.

Die folgenden Ausführungen beziehen sich der Einfachheit halber auf ein System 1. Ordnung. Der zugrundeliegende Algorithmus spannt zunächst einen Zustandsgitter $X^{N\times m}$ auf, wobei N die Länge des Problems und m die Anzahl der Sütztstellen im Zustandsbereich $x_{min} - x_{max}$ ist. Analog enthält $U^{N\times m}$ das Stellgrößengitter entlang des Problems mit einer Diskretisierung von p Stützstellen. Ausgehend vom Endzustand x_N wird für jede Stützstelle in $X^{N\times m}$ die Kostenfunktion (3.3) mit den möglichen Stellgrößen bestimmt und die Stützstelle mit dem niedrigsten Kostenfunktionswert für den nächsten Zustand herangezogen. Ist der Algorithmus beim Anfangszustand x_0 angelangt, wurde die optimale Steuerfolge π^O an den diskreten Stützstellen bestimmt. Da das System (3.1) auch Zustände und Stellgrößen zwischen diesen Stützstellen erlaubt, erfolgt im Anschluss eine Vorwärtssimulation, wobei mittels linearer Interpolation zwischen den Stützstellen approximiert wird.

Tabelle 3.2 zeigt die Parameter und ihre Beschreibung die für die Anwendung der Funktion dpm.m in dieser Arbeit relevant sind.

Da die dynamische Programmierung ein sehr rechenintensives Optimierungsverfahren ist, müssen Maßnahmen ergriffen werden, die eine Reduktion der Rechenzeit bewirken können. Dabei müssen eventuelle Auswirkungen auf die Güte der Optimierung berücksichtigt werden. Im folgenden werden einige dieser Möglichkeiten und deren Auswirkung zusammengefasst.

Reduktion der Modellordnung: Die Rechenzeit steigt bei der dynamischen Programmierung mit einer Erhöhung der Modellordnung exponentiell. Aus diesem Grund sollten im Modell nur die notwendigsten Zustände dynamisch modelliert werden. Für die verbleibenden Zustände bietet sich gegebenfalls die quasistatische Modellbildung an.

Schrittweite: Die Schrittweite im Modell definiert die Problemlänge und hat damit direkten Einfluss auf die Rechenzeit. Die Erhöhung der Schrittweite führt jedoch zu einem ungenaueren Modell und reduziert damit die Güte der Optimierung.

Parameter	Beschreibung
`Ts`	Fixe Schrittweite im Modell
`N`	Problemlänge
`W{.}`	Vektoren der Länge N, enthalten schrittvariante Daten des Modells
`Nx{.}`	Anzahl der Stützstellen im Zustandsraum (m)
`Xn{.}.lo`	Untere Grenze x_{min} des Zustands
`Xn{.}.hi`	Obere Grenze x_{max} des Zustands
`XN{.}.lo`	Untere Grenze $x_{N,min}$ des Endzustands
`XN{.}.hi`	Obere Grenze $x_{N,max}$ des Endzustands
`X0{.}`	Anfangszustand x_0
`Nu{.}`	Anzahl der Stützstellen der Stellgröße (p)
`Un{.}.lo`	Untere Grenze u_{min} der Stellgröße
`Un{.}.hi`	Obere Grenze u_{max} der Stellgröße
`UseLine`	*Boundary Line Method* - Methode zur Einschränkung des Zustandsraums vor der Optimierung
`InfCost`	Unendlichkeitswert für die Bestrafung von unzulässigen Zustandsübergängen

Tabelle 3.2: Parameter der Funktion `dpm.m`

Diskretisierung: Mehr Stützstellen im Zustandsraum und im Stellgrößenvektor bedeuten mehr Rechenzeit, führen aber durch eine feinere Interpolation zu einer besseren Güte.

Begrenzen des Zustands: Ist der Bereich bekannt und abgrenzbar, indem sich der Zustand bewegt, kann man durch das Setzen von unteren und oberen Zustandsgrenzen eine bessere Güte bei gleichbleibender Anzahl von Stützstellen in diesem Bereich erreicht werden, da in diesem Fall mit gleichbleibender Anzahl von Stützstellen dieser Bereich besser diskretisiert werden kann.

Ausschluss von unzulässigen Zustandsübergängen: Durch geeignete Methoden (*Boundary Line Method*) können im Vorfeld der Optimierung Zustandsübergänge und Zustandsstützstellen ausgeschlossen werden, die bei bekanntem Anfangs- und Endzustand und definiertem Stellgrößenbereich nicht erreichbar sind. Die so ausgeschlossenen Stützstellen müssen bei der Optimierung nicht mehr berücksichtigt werden, was bei einer großen Problemlänge den Rechenaufwand für die Vorberechnung rechtfertigt.

3.3 Einsatz der dynamischen Programmierung beim Fahrzeugmodell

Die Aufgabe für die Optimierung bei einem Serienhybrid besteht darin, den Verbrauch der Verbrennungskraftmaschine des Range Extenders zu minimieren und dabei den gleichen Endladungszustand der Batterie zu erreichen, der am Anfang gegeben war. Dabei muss der Einfluss der Steigungen der Strecke mitberücksichtigt werden, da diese eine besondere Last oder eine Möglichkeit zur Rekuperation darstellen. Für die Berechnung dieser Steuerfolge mittels der im Abschnitt 3.2 vorgestellten MATLAB®-Funktion `dpm.m` muss das komplette Fahrzeugmodell basierend auf den im Kapitel 2 beschriebenen Komponenten eingebunden werden. Dabei gehen Geschwindigkeits- und Steigungsprofil der Strecke als Störungen in das Modell ein, die während der Optimierung mitberücksichtigt werden müssen.

Das für die Optimierung implementierte System wird in Abbildung 3.3 dargestellt. Dabei wird ausgehend vom bekannten Geschwindigkeits- und Höhenprofil das Antriebsmoment

und die Winkelgeschwindigkeit der Räder wie im Abschnitt 2.2.1 bestimmt. Dieses Moment wird über ein Getriebe in eine Momentenanforderung M_{EM} und in eine Winkelgeschwindigkeitsanforderung ω_{EM} an der elektrischen Maschine umgewandelt, die zu einer elektrischen Leistungsaufnahme P_{EM} bei motorischem Betrieb oder einer Leistungsabgabe im generatorischen Betrieb nach Abschnitt 2.2.3 führt. Auf der anderen Seite der Wirkkette befindet sich der Range Extender, dessen Betriebszustand *ein* oder *aus* durch die Stellgröße μ_k gegeben ist. Für den eingeschalteten Zustand der Verbrennungskraftmaschine ist ein Drehmoment M_{soll} und eine Winkelgeschwindigkeit ω_{soll} gegeben, die einen Betriebsbereich mit hohem Wirkungsgrad definieren. Wie in Abschnitt 2.2.5 beschrieben, resultiert diese Betriebsanforderung in einer thermischen Leistungsaufnahme P_{ICE}, die dem Kraftstoffverbrauch und somit den Kosten entspricht. Der elektrische Generator erhält die durch ein Getriebe umgewandelte Drehmoment- und Winkelgeschwindigkeitsanforderung von der Verbrennungskraftmaschine und gibt die entsprechende elektrische Leistung P_{EG} an die Batterie ab. Die Leistungsbilanz der Batterie P_{BT} setzt sich aus der Summe der elektrischen Leistung der elektrischen Maschine P_{EM} und des Generators P_{EG} zusammen und bewirkt wie in Abschnitt 2.2.6 beschrieben eine Änderung der Ladung SOC.

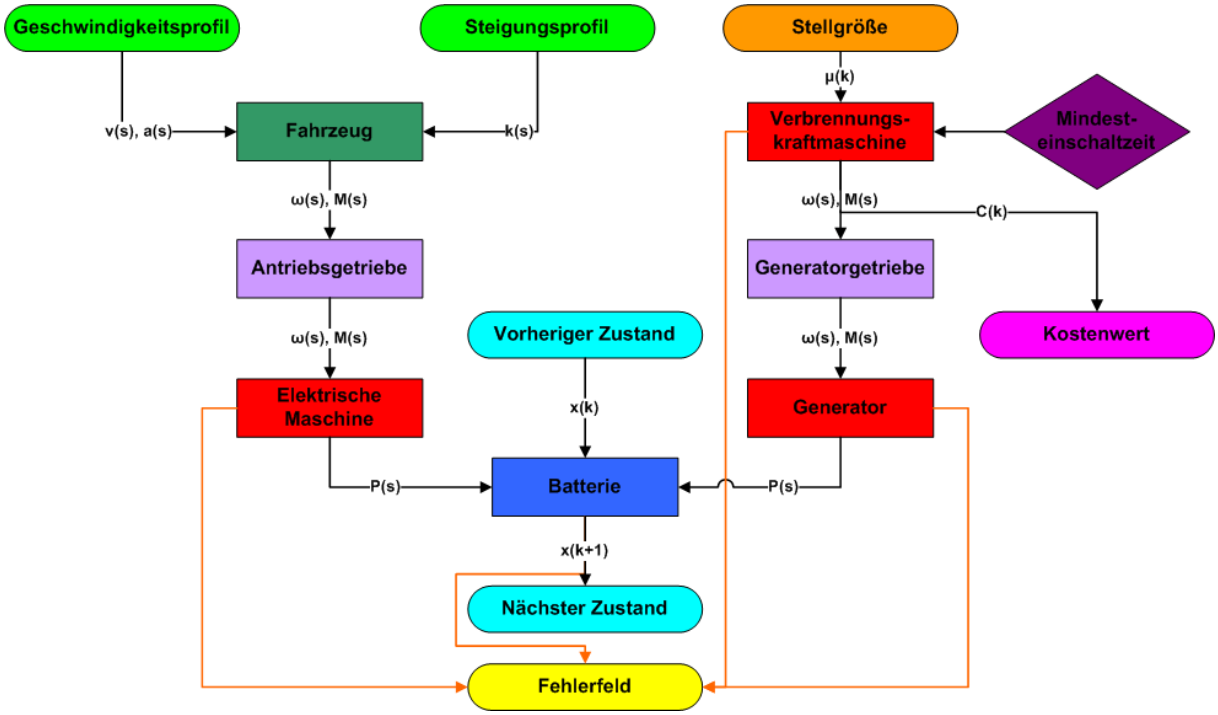


Abbildung 3.3: Systemüberblick Serienhybrid

Die Stellgröße μ_k ist im Fall eines Serienhybrids mit Range Extender ein Schalter mit zwei Stellungen: entweder ist die Verbrennungskraftmaschine eingeschaltet und lädt über den Generator die Batterie, oder sie ist ausgeschaltet und die Batterie erhält keine Unterstützung für die Versorgung des Fahrzeugs. Im ersten Schritt der Optimierung muss der Algorithmus ausgehend vom bekannten Endzustand x_{N-1} in Abhängigkeit davon, ob die Verbrennungskraftmaschine eingeschaltet war oder nicht, feststellen, welche zwei Zustände als der nächste Zustand x_{N-2} möglich sind und welche Kosten C_{N-1} mit dem Übergang in diese Zustände verbunden sind. Ist der Algorithmus beim Anfangszustand x_0 angelangt, wurde eine Steuerfolge $\pi^O = \mu_0, \mu_1, ..., \mu_{N-1}$ gefunden, die unter minimalen Kosten C_π das Ziel der Optimierung, nämlich den Endzustand x_{N-1} gleich dem Anfangszustand x_0 mit der geforderten Toleranz zu

setzen, erreicht.

Die Wahl der Stellgröße μ_k durch den Algorithmus kann dabei im Modell durch eine Forderung an die Mindesteinschaltzeit überschrieben werden. So eine Forderung ist notwendig, um unrealistisch kurze Einschaltimpulse der Verbrennungskraftmaschine zu unterbinden. Außerdem wird bei der Berechnung des Kostenwerts C_k ein Einschaltvorgang der Verbrennungskraftmaschine mit einem zusätzlichen Faktor bestraft. Dadurch wird während der Optimierung eine Mindesteinschaltdauer erzwungen und in der Kostenfunktion berücksichtigt, um das gewünschte Optimierungsergebnis zu erhalten.

Eine weitere Nebenbedingung während der Optimierung ist wie in Kapitel 2 beschrieben die Einhaltung gewisser Betriebsgrenzen in den einzelnen Komponenten. Wird durch eine Komponente eine Überschreitung einer solchen Grenze erkannt, wird für den entsprechenden Zustand x_k und die Stellgröße u_k ein Eintrag im Fehlerfeld Inf_k gemacht. Bei der Auswertung der Kostenfunktion C_k durch die Funktion `dpm.m` werden Kostenwerte, die zu einer Zustands- und Stellgrößenkombination gehören, bei denen ein von Null verschiedener Eintrag im Fehlerfeld vorhanden ist, auf den vom Optimierungsalgorithmus als Unendlich interpretierten Wert gesetzt. Damit werden solche Zustands- und Stellgrößenkombinationen für die Optimierung als unzulässig markiert.

3.3.1 Parameter des Fahrzeugmodells

Um im Anschluss an die Optimierung eine Evaluierung der Ergebnisse mit Hilfe der Software AVL Cruise® durchführen zu können, wurden die Parameter (Tabelle 3.3) des Fahrzeugs aus dem Beispielmodell *Range Extender* von AVL Cruise® übernommen. Dieses Modell entspricht dem Aufbau eines Serienhybrids bestehend aus einem Antrieb mit einer elektrischen Maschine und einem benzinbetriebenen Range Extender (Abbildung 3.4).

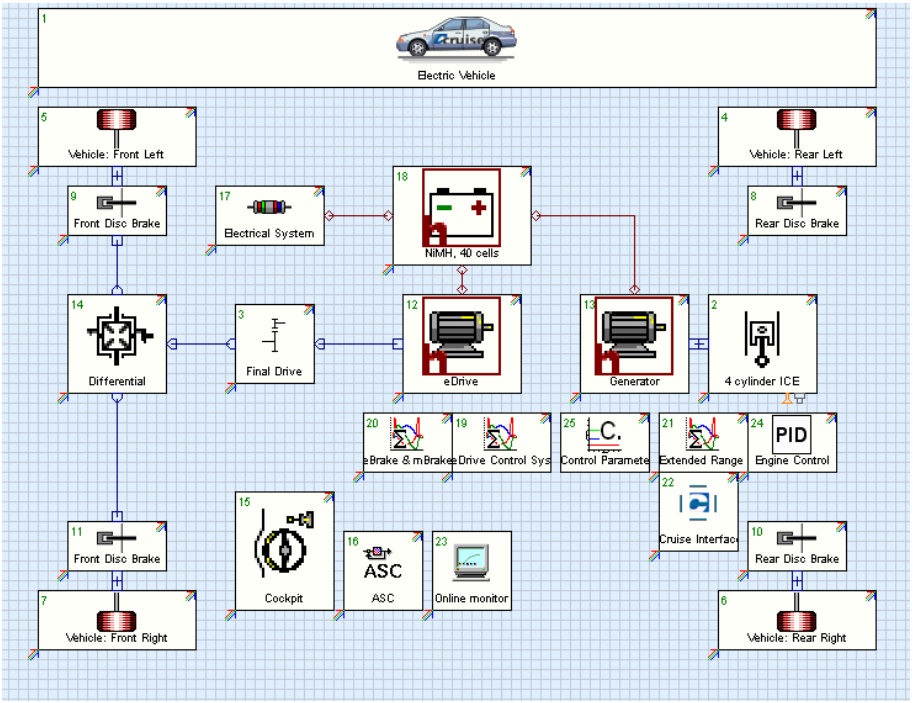


Abbildung 3.4: Systemüberblick AVL Cruise®-Modell eines Serienhybridfahrzeugs

Im Gegensatz zu der im Kapitel 2 vorgestellten Modellierung werden im AVL Cruise®-Modell transiente Übergänge auch mitberechnet, wie zum Beispiel der geregelte Hochlauf des Range Extender Blocks bestehend aus Benzinmotor und elektrischem Generator. Die vom Fahrer geforderte Bremswirkung wird nur zu dem Teil rekuperiert, wie das vom Ansprechverhalten der elektrischen Maschine möglich ist, die restliche Energie wird mittels mechanischem Bremsen in Wärme umgewandelt. AVL Cruise® bietet eine Schnittstelle zu MATLAB®, womit dann im späteren Verlauf ein direkter Vergleich der Ergebnisse aus den beiden Modellen möglich wird.

Parameter	**Wert**
Radradius	$0.3\,[m]$
Rotierende Masse	$5\,[\%]$
Fahrzeugmasse	$1200\,[kg]$
Angriffsfläche	$1.97\,\left[m^2\right]$
cw-Wert	$0.284\,[-]$
Getriebeübersetzung Antrieb	$6.058\,[-]$
Wirkungsgrad	$0.98\,[-]$
Skalierungsfaktor elektrische Maschine	$3.43\,[-]$
Trägheitsmoment elektrische Maschine	$10^{-4}\,\left[kg \cdot m^2\right]$
Zusatzverbraucher	$300\,[W]$
Skalierungsfaktor Generator	$3.43\,[-]$
Unterer Krafstoffheizwert	$42.7 \cdot 10^6\,\left[\frac{J}{kg}\right]$
Hub	$2\,\left[dm^3\right]$
Skalierungsfaktor Verbrennungskraftmaschine	$2\,[-]$
Trägheitsmoment Verbrennungskraftmaschine	$0.134\,\left[kg \cdot m^2\right]$
Stellwinkelgeschwindigkeit	$315\,\left[\frac{rad}{s}\right]$
Nennspannung Batterie	$280\,[V]$
Maximale Batterieladung	$20\,[Ah]$
Innenwiderstand Entladen	$1\,[\Omega]\,@70\%$
Innenwiderstand Laden	$0.8\,[\Omega]\,@70\%$
Wirkungsgrad Entladen	$0.9\,[-]$
Wirkungsgrad Laden	$0.85\,[-]$

Tabelle 3.3: Parameter des Fahrzeugmodells

3.3.2 Parameter der Optimierung

In diesem Abschnitt werden die Optimierungsparameter für die Bestimmung einer optimalen Steuerfolge des Range Extenders für das im Abschnitt 3.3.1 vorgestellte Fahrzeugmodell beschrieben. Wie im Abschnitt 3.2 bereits diskutiert, beeinflussen einige dieser Parameter die Rechenzeit und die Güte der Optimierung maßgeblich. Durch die Variation dieser Parameter wird in diesem Abschnitt ein Vergleich ihrer Auswirkungen möglich und es kann ein Kompromiss für den weiteren Verlauf dieser Arbeit gefunden werden.

Die statischen Parameter aus Tabelle 3.2, die keinen Einfluss auf die Güte und die Rechenzeit haben, sind die Störvektoren `W{.}`, die den Geschwindigkeits- und Steigungsverlauf der Strecke beinhalten, der Anfangszustand `X0{.}`, sowie die Grenzen des Zustandsraums `Xn{.}.lo` und `Xn{.}.hi` und des Endzustands `XN{.}.lo` und `XN{.}.hi`. Da die Steuerfolge π als eine Schaltfolge für die Verbrennungskraftmaschine definiert ist, sind die Anzahl

der Stützstellen `Nu{.}` und die Grenzen der Steuergröße `Un{.}lo` und `Un{.}.hi` auch fixiert. Die Werte dieser Parameter sind aus Tabelle 3.4 zu entnehmen.

Der Unendlichkeitswert `InfCost` muss so gewählt werden, dass er immer größer ist, wie der maximal zulässige Kostenwert C_{max} für einen Schritt, jedoch diesen nicht um mehrere Größenordnungen überschreitet, da dies zu numerischen Fehlern bei der Berechnung führen kann. Da die Kostenfunktion C^{π} dem Kraftstoffmassendurchfluss der Verbrennungskraftmaschine nach Gleichung 3.5 entspricht, wurde `InfCost = 2` gewählt.

$$C^{\pi} = \frac{P_{ICE}}{H_u} \cdot \frac{\Delta s}{v} \tag{3.5}$$

Da bei Systemen erster Ordnung und bei Zustandsraumgrößen und Problemlängen, die in dieser Arbeit vorkommen, die Verwendung der *Boundary Line Method* zur Vorberechnung von unerreichbaren Systemzuständen die gleiche Zeit in Anspruch nimmt, wie die Optimierung ohne *Boundary Line Method*, wird diese nicht verwendet.

`W{1} = v_of_s`
`W{2} = dv_of_s`
`W{3} = incl_of_s`
`X0{1} = 0.7`
`Xn{1}.lo = 0.2`
`Xn{1}.hi = 0.9`
`XN{1}.lo = 0.99·X0`
`XN{1}.hi = 1.01·X0`
`Nu{1} = 2`
`Un{1}.lo = 0`
`Un{1}.hi = 1`
`InfCost = 2`
`UseLine = 0`

Tabelle 3.4: Statische Parameter der Optimierung

Die verbleibenden Parameter Problemlänge `N`, Schrittweite `Ts` und Anzahl der Stützstellen im Zustandsraum `Nx{.}` beeinflussen die Güte und die Rechenzeit der Optimierung maßgeblich. Um für diese Parameter einen guten Kompromiss finden zu können, wurden mehrere Optimierungsläufe mit verschiedenen Parameterkombinationen durchgeführt. Diese werden in Tabelle 3.5 dargestellt. Für jede dieser Kombinationen wurde die Güte als Abweichung von der Zielladung von 70% mit dem Zusammenhang $\left(1 - \frac{x_0}{x_N}\right) \cdot 100$ berechnet, wobei in der Optimierung eine Toleranz von 1% vorgegeben worden ist. Die gemessene Rechenzeit basiert auf einer Optimierung mit einem Microsoft® Windows XP 32-bit PC mit einem Intel® T2500 CPU 2.00 GHz und 2.0 GB RAM.

Anhand der Ergebnisse aus Tabelle 3.5 ist erkennbar, dass die Problemlänge den größten Einfluss auf die Güte und die Rechenzeit besitzt. Die Rechenzeit erhöht sich proportional der Prob-lemlänge, jedoch ist dieser Zusammenhang bei der Verbesserung der Güte bei einer feineren Schrittweite nicht so stark ausgeprägt. Der Einfluss der Anzahl der Stützstellen kommt weit weniger zum Tragen. Es ist lediglich ein kleiner Unterschied in der Rechenzeit feststellbar, eine Aussage über die Güte ist nicht möglich, da hier die Unterschiede in der Toleranz der Endladung auch mitberücksichtigt werden müssen. Es ist jedoch festzuhalten, dass mit Unterschreitung einer gewissen Anzahl an Stützstellen eine Optimierung nicht mehr möglich ist.

Ts $[m]$	**N**	**Nx**	**Rechenzeit** $[s]$	**C** $\left[\frac{l}{100km}\right]$	**Endladung** $[\%]$	**Abweichung** $[\%]$
20	944	51	56,91	4,627	70,16	0,22
20	944	71	57,25	4,621	70,09	0,12
20	944	151	59,66	4,62	70,3	0,42
50	378	51	20,85	4,742	70,07	0,09
50	378	71	21,83	4,743	70,45	0,63
50	378	151	21,39	4,682	69,75	-0,35
100	189	51	10,53	5,061	70	0
100	189	71	10,41	5,003	70,02	0,07
100	189	151	10,97	4,993	70,05	0,02
250	76	51	4,65	5,097	70,39	0,55
250	76	71	4,61	5,066	70,44	0,62
250	76	151	4,44	5,009	69,81	-0,27
500	38	51	2,28	6,116	74,08	5,5
500	38	71	2,27	5,922	69,14	-1,24
500	38	151	2,27	6,257	70,82	1,15

Tabelle 3.5: Vergleich der Parameterkombinationen der Optimierung

In diesem Fall ist diese Grenze knapp unter `Nx = 50`, wobei für eine stabile Funktion in allen Bereichen eine Wahl von `Nx = 71` getroffen worden ist. Damit entspricht die Differenz zwischen zwei Stützstellen genau 1% Gesamtladungsunterschied.

Wie für die Anzahl der Stützstellen, gelten für die minimale Problemlänge auch Grenzen. Es ist gut erkennbar, dass mit der Wahl von `Ts = 250` die Abweichung in der Endladung im Schnitt höher wird und bei `Ts = 500` sogar die erlaubte Toleranz von 1% in der Zielladung überschreitet. Weiters sind damit auch erhöhte Kosten verbunden.

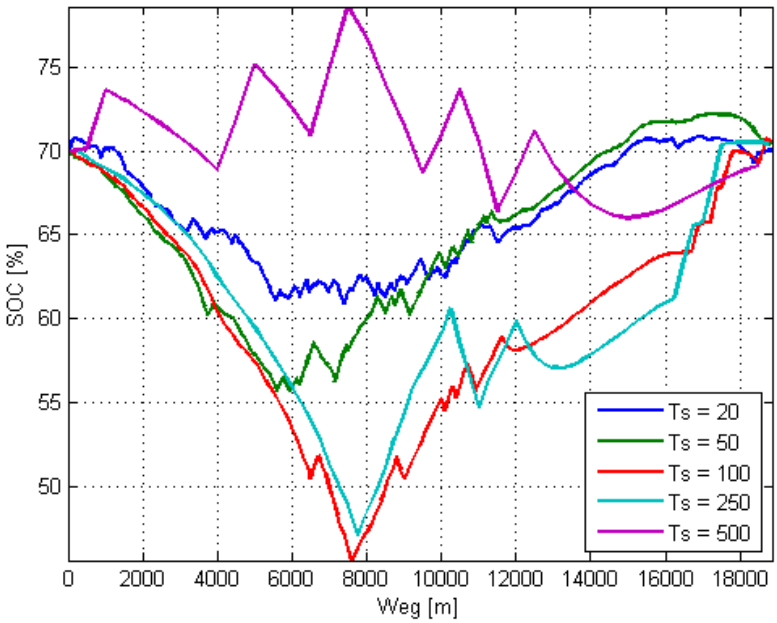


Abbildung 3.5: Vergleich der Ladungsverläufe bei verschiedenen Schrittweiten

Wie in Abbildung 3.5 zu sehen ist, hat die Wahl der Schrittweite einen entscheidenden Einfluss auf den Ladungsverlauf. Da eine Vorhersage des Streckenprofils mit einer Genauigkeit von `Ts = 20` oder `Ts = 50` unrealistisch ist und die Berechnung einer Steuerung von Haus aus

nicht auf Änderungen von Störgrößen reagieren kann, wurde bei dieser Arbeit die Schrittweite von `Ts = 100` gewählt. Der daraus resultierenden Ungenauigkeit im Geschwindigkeits- und Steigungsprofil soll mit Hilfe des im Kapitel 4 vorgestellten modellprädiktiven Reglers entgegengewirkt werden.

3.3.3 Ergebnisse der Optimierung bei den gewählten Streckenverläufen

In diesem Abschnitt werden die Ergebnisse der Optimierung für alle im Abschnitt 2.4 vorgestellten Strecken vorgestellt, um die Plausibilität der gewählten Parameter aus Abschnitt 3.3.2 bei verschiedenen Strecken zu überprüfen. Diese sind in den Abbildungen 3.6-3.8 dargestellt. Alle Abbildungen enthalten die wichtigsten Verläufe: Geschwindigkeits- und Steigungsprofil, Batterieladung, Batteriestrom und die Leistungsdaten der Energiewandler.

Wie man in den Abbildungen 3.6-3.8 erkennt, funktioniert die Optimierung mit dem gewählten Parametersatz für alle drei Strecken. Tabelle 3.6 zeigt die Kosten, Rechenzeit, Problemlänge und Abweichung bei der Zielladung für diese Strecken. Dabei ist anzumerken, dass bei der Budapest-Strecke eine Schrittweite von `Ts = 50` gewählt worden ist, um den Anforderungen einer Stadtfahrt besser gerecht zu werden.

Strecke	**Ts** $[m]$	**N**	**Nx**	**Rechenzeit** $[s]$	**C** $\left[\frac{l}{100km}\right]$	**Endladung** $[\%]$	**Abweichung** $[\%]$
Rechberg	100	189	71	10,41	5,003	70,05	0,07
Budapest	50	427	71	36,06	2,413	69,39	-0,87
Graz-Wolfsberg	100	335	71	19,82	3,337	69,96	-0,05

Tabelle 3.6: Vergleich der Parameterkombinationen der Optimierung

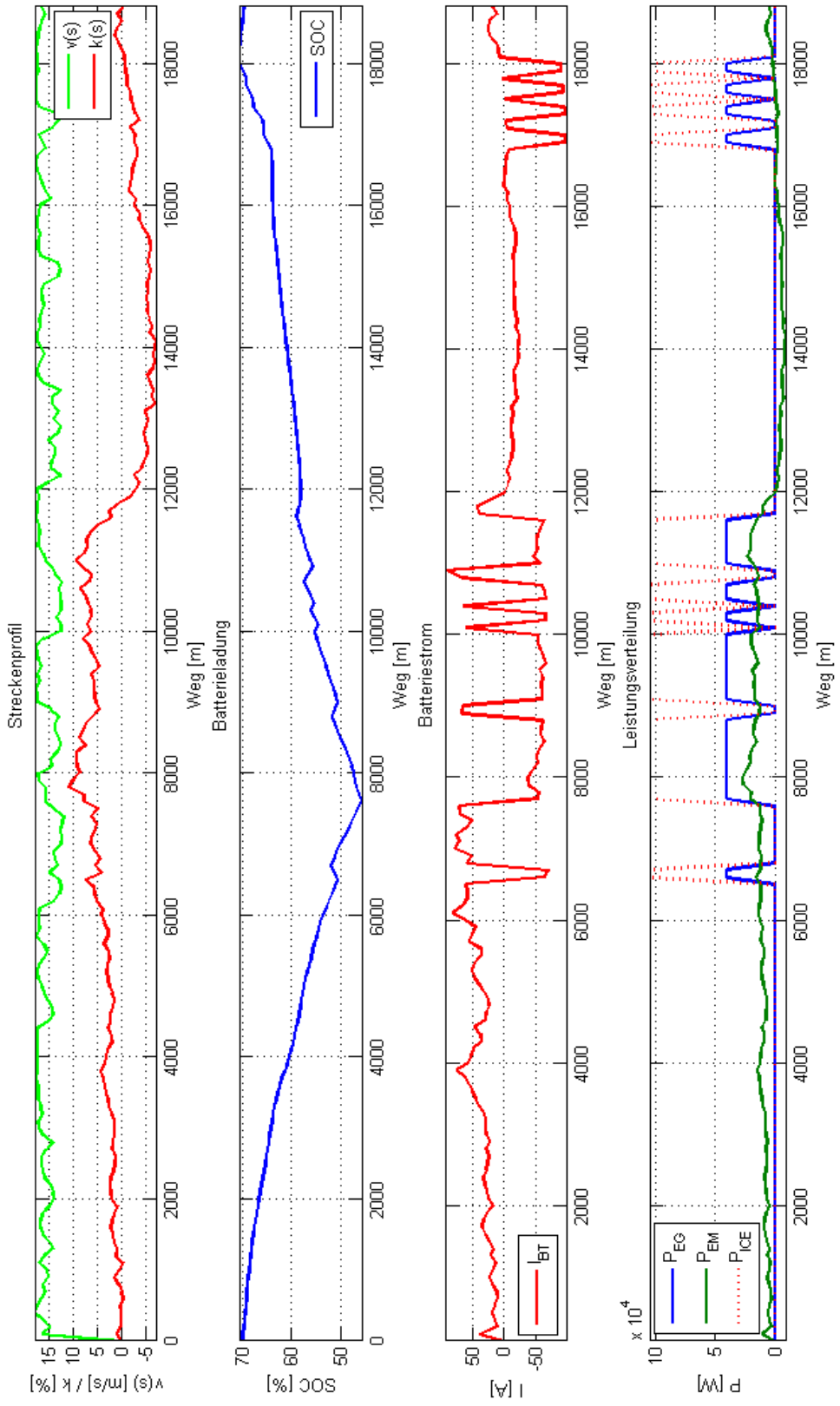


Abbildung 3.6: Verlauf der Optimierung an der Rechberg Strecke

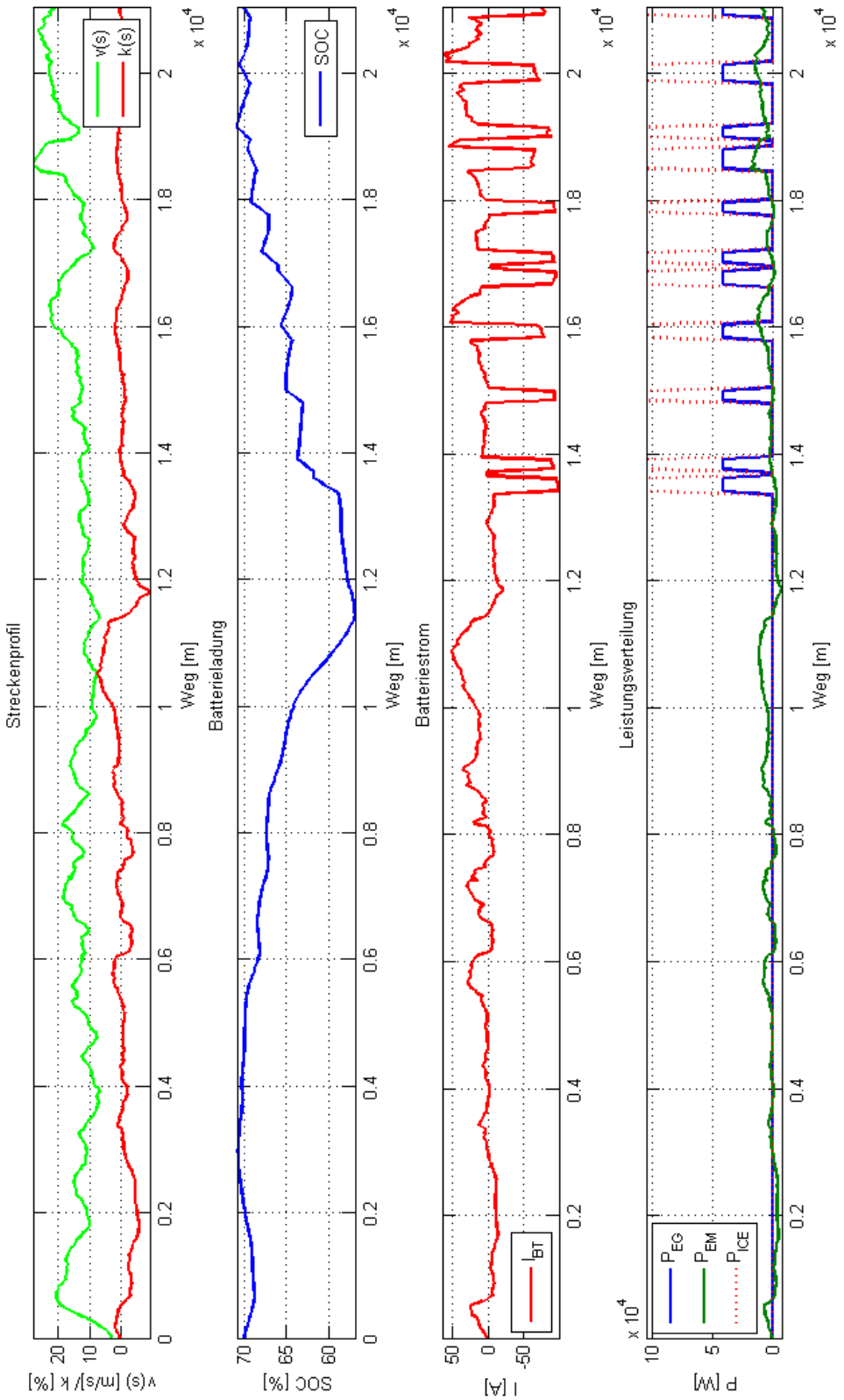


Abbildung 3.7: Verlauf der Optimierung an der Budapest Strecke

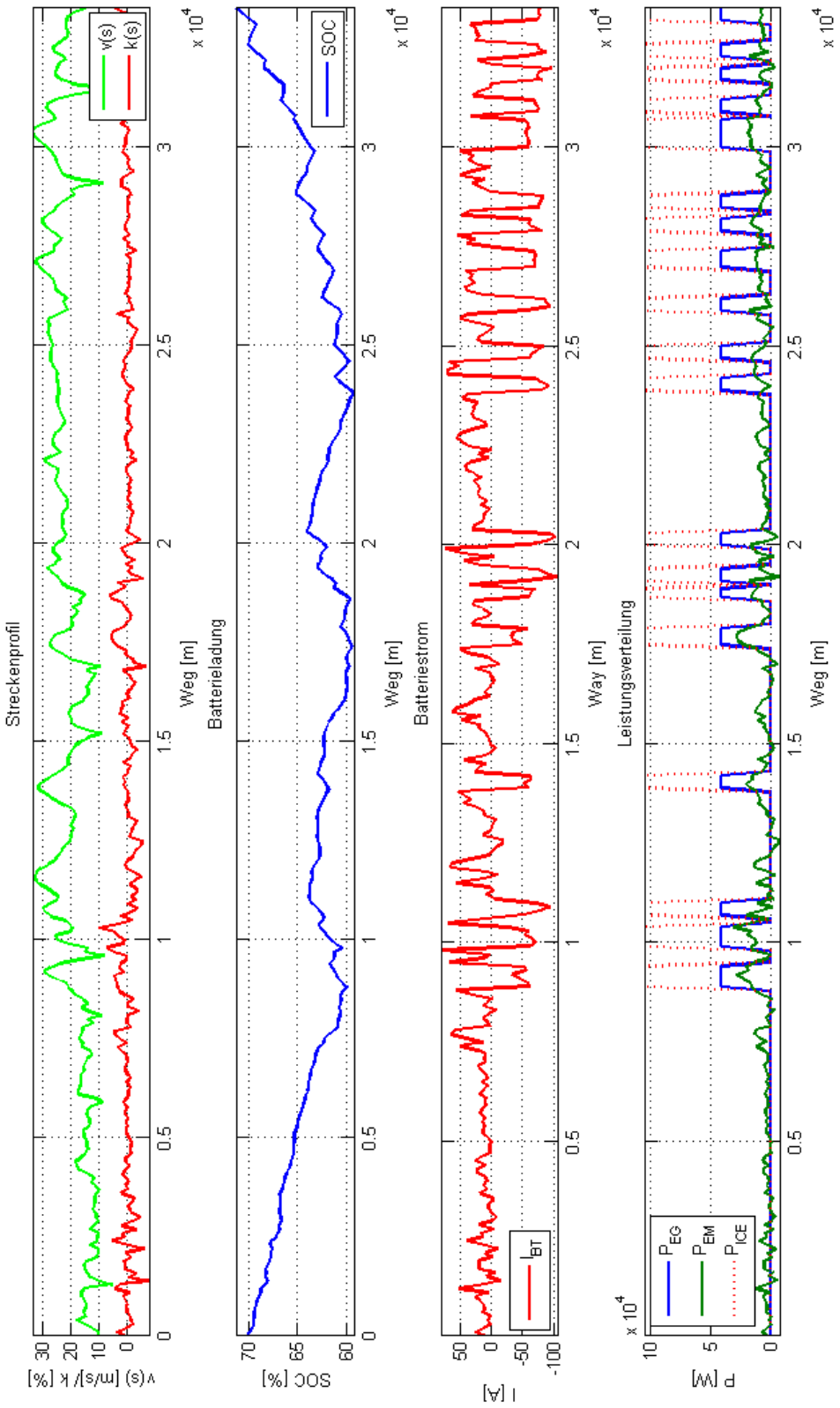


Abbildung 3.8: Verlauf der Optimierung an der Graz-Wolfsberg Strecke

Kapitel 4

Modellprädiktive Regelung

Da die vorhin beschriebene Optimierungsmethode lediglich eine optimale Steuerung für das prädizierte Fahrprofil gibt, ist dieser Lösungsweg in der Realität zum Scheitern verurteilt. Einerseits wird die Vorhersage des Geschwindigkeitsprofils entlang einer bekannter Strecke sehr stark durch Störungen aus der Umgebung wie der Verkehrssituation, Ampelschaltungen, verschiedenen Wartezeiten an Kreuzungen, Überholmanövern und anderen Faktoren erschwert, andererseits hat der Fahrer immer die Möglichkeit, Entscheidungen zu treffen, die nicht zu prädizieren sind, wie zum Beispiel eine Änderung der Fahrstrecke. Daraus folgt, dass je länger die prädizierte Fahrstrecke ist, desto größere Abweichungen werden sich zwischen prädiziertem und tatsäch-lichem Fahrprofil ergeben, die von einer optimieren Steuerung nicht mehr berücksichtigt werden können. Daher ist die Verwendung einer optimalen Steuerung für die Lösung der Aufgabe in der Realität unpraktikabel.

Abhilfe würde ein Regler schaffen, der bei jedem Problemschritt die Optimierung neu ausführt und dabei den aktuellen, gemessenen Zustand des realen Fahrzeugs als Anfangszustand für die neue Optimierung annimmt. Diese Vorgangsweise hat den Nachteil, dass sie bei großen Problemlängen einen sehr hohen Rechenaufwand hat, da durchschnittlich eine $\frac{N}{2}$ lange Optimierung N-mal ausgeführt werden muss. Dabei ist damit zu rechnen, dass die in der Zukunft liegenden Zustände durch die ungenaue Prädiktion immer ungenauer werden, wodurch der Kosten-Nutzen-Faktor für die Berechnung schlecht wird.

Bei der Idee der *modellprädiktiven Regelung* wird diese Vorgangsweise abgewandelt und es werden nur eine begrenzte Anzahl von zukünftigen Zuständen für jeden Optimierungsschritt betrachtet. Diese Idee wird in den Arbeiten von Wang (2009), Stiegler (2008) und Sundström (2009) beschrieben. Damit lässt sich ein guter Kompromiss zwischen Rechenzeit und Güte erzielen.

4.1 Theorie der modellprädiktiven Regelung

In Abbildung 4.1 ist der grundlegende Ablauf einer modellprädiktiven Regelung[1] dargestellt. Im Schritt k wird der aktuelle Zustand gemessen und eine Optimierung mit der dem Prädiktionshorizont entsprechender Länge ausgeführt. Dabei ist das Ziel der Optimierung entlang des

[1] Abbildung überarbeitet aus Wikipedia http://en.wikipedia.org/wiki/File:MPC_scheme_basic.svg

Prädiktionshorizonts mit der Referenztrajektorie gegeben. Das Ergebnis der Optimierung liefert zu jedem Schritt k eine optimale Steuerfolge π_k^O für den Prädiktionshorizont. Aus dieser Steuerfolge π_k^O wird das erste Element $\mu_0 \in \pi_k^O$ auf das System aufgeschaltet. Im Zeitschritt $k+1$ wird der Prädiktionshorizont um einen Schritt nach vorne geschoben und die Optimierung erneut durchgeführt.

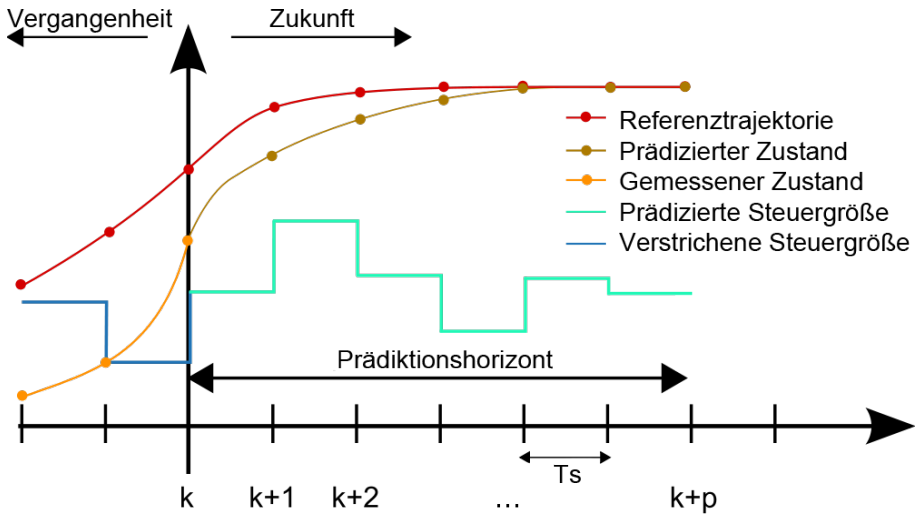


Abbildung 4.1: Ablauf der modellprädiktiven Regelung

Durch diese Vorgangsweise wir der aktuelle, tatsächliche Zustand des Systems und damit während der Regelung auftretenden Störungen mitberücksichtigt. Der daraus resultierende Aufbau des Regelsystems ist in Abbildung 4.2 dargestellt.

Mathematisch lässt sich der Regler in zwei Schritten modellieren: Gleichung 4.1 zeigt die Berechnung der Steuerfolge ausgehend vom aktuellen Zustand mittels dynamischer Programmierung, in Gleichung 4.2 wird das erste Element der Steuerfolge auf die Strecke angewandt. Dabei ist die Länge des Prädiktionshorizonts mit p und die Referenztrajektorie mit r_k benannt. Die optimale Steuerfolge entsteht bei einer Problemlänge N nach Gleichung 4.3 aus den auf die Strecke geschalteten Steuergrößen.

$$C = min\left(\sum_{i=0}^{p} \hat{\mu}_i\right), \quad \hat{x}_0 = x_k, \quad \hat{x}_p = r_{k+p} \tag{4.1}$$

$$x_{k+1} = F\left(x_k, \hat{\mu}_{0,k}\right) \tag{4.2}$$

$$\pi^O = \{\hat{\mu}_{0,0}, \hat{\mu}_{0,1}, ..., \hat{\mu}_{0,N-1}\} \tag{4.3}$$

Die Wahl der Länge des Prädiktionshorizonts ist für diese Art von Reglern entscheidend: wählt man den Prädiktionshorizont zu kurz, hat der Regler eine eingeschränkte Möglichkeit, den Zustandsraum vollständig auszunutzen, da ihm weniger Schritte zum Erreichen des Endzustans zur Verfügung stehen. Wählt man den Prädiktionshorizont jedoch zu lang, steigt die Rechenzeit stark an, wobei der Nutzen dieses Anstiegs nicht gerechtfertigt werden kann, da weit in der Zukunft liegende prädizierte Zustände mit immer größeren Unsicherheiten behaftet sind.

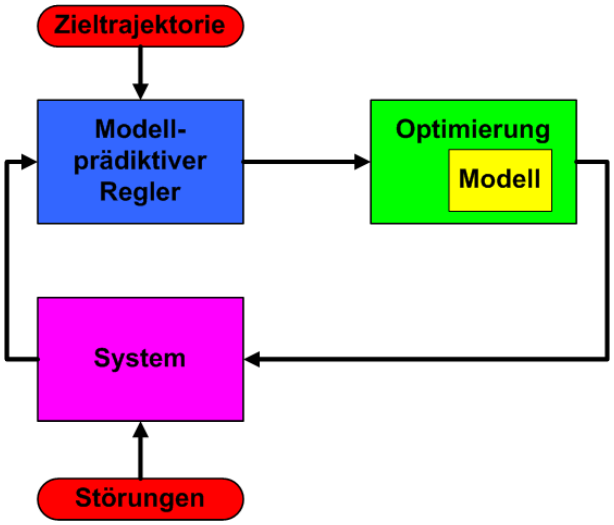


Abbildung 4.2: Aufbau eines modellprädiktiven Regelsystems

4.2 Anwendung der modellprädiktiven Regelung auf das Fahrzeugmodell

In diesem Abschnitt wird die Einbettung des Fahrzeugsmodells in eine Reglerstruktur aus Abbildung 4.2, wie es im Abschnitt 4.1 beschrieben worden ist, vorgestellt.

Im Fall des Fahrzeugmodells müssen folgende Größen definiert werden:

- Die Länge des gewählten Prädiktionshorizonts in Abhängigkeit der auftretenden Störungen
- Die Referenztrajektorie entlang der betrachteten Strecke

4.2.1 Bestimmung des Prädiktionshorizonts

Die Länge des Prädiktionshorizonts wurde nach folgender Überlegung auf den Wert $s_p = 1\ km$ entsprechend einer Optimierungslänge von $n_p = 10$ bei einer Streckendiskretierung von $\Delta s = 100\ m$ gesetzt: die Geschwindigkeit eines Fahrzeugs lässt sich auch bei einer Landstraßenfahrt nicht weiter wie einige hundert Meter bestimmen, außerdem würde ein längerer Prädiktionshorizont die Rechenzeit des Reglers unnötig verlängern.

Diese Überlegung lässt sich an folgendem kurzem Beispiel veranschaulichen: sei die Länge der Strecke mit $10\ km$ und die Streckenauflösung mit $100\ m$ gegeben. In diesem Fall beträgt die Länge des Problems $N = 100$. Wählt man die Länge des Prädiktionshorizonts mit $1\ km$, erhält man für die Problemlänge *eines* Optimierungsdurchlaufs der Regelung für das Prädiktionshorizont $n_p = 10$. Da diese Optimierung mit der Länge n_p bei jedem Streckenpunkt durchgeführt wird, ist die Rechenzeit t mit $N \cdot n_p = 10000$ proportional. Wird die Länge des Prädiktionshorizonts verdoppelt, verdoppelt sich dementsprechend auch die Rechenzeit, ohne eine bessere Güte zu bewirken, da die Berechnung der Optimierung entlang des Prädiktionshorizonts zwar vom bekannten Anfangszustand ausgeht, aber für den akutellen Optimierungsdurchlauf keine weiteren Störungen mitberücksichtigt werden können.

4.2.2 Bestimmung der Referenztrajektorie

Die Wahl der Referenztrajektorie richtet sich nach dem Ziel der Optimierung: es soll der gleiche Ladungszustand der Batterie am Ende der Fahrt erreicht werden, wie er am Anfang gegeben war. Die einfachste Überlegung wäre, die Referenztrajektorie r_k entlang der gesamten Strecke auf die gewünschte Endladung x_N zu setzten. Diese Vorgangsweise ist aber durch die Tatsache, dass nur eine begrenzte Teilstreckenlänge für die einzelnen Optimierungsschritte betrachtet wird, nicht vorteilhaft, da der Regler dadurch in einer viel kürzeren Zeitspanne den gewünschten Endladungszustand erreichen muss. Dadurch würde der Regler ständig versuchen, einer Entladung der Batterie entgegenzuwirken und den zur Verfügung stehenden zulässigen Ladungsbereich nur schlecht ausnutzen.

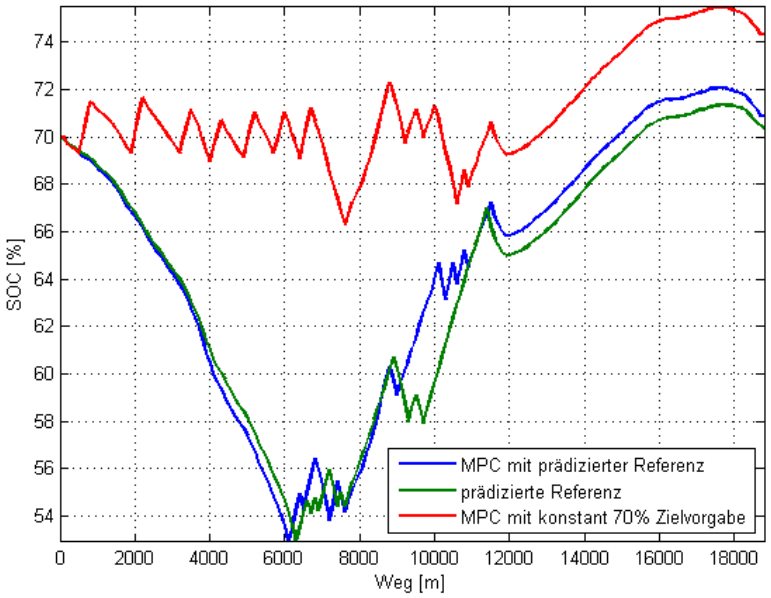


Abbildung 4.3: Vergleich der Ladungsverläufe der modellprädiktiven Regelung mit und ohne Prädiktion des Ladungsverlaufs für die Rechberg-Strecke

Dieses Verhalten ist in Abbildung 4.3 dargestellt. Der rote Verlauf zeigt die Ladung mit einer Referenzvorgabe von $70\ \%$ entlang der gesamten Strecke. Der Regler versucht, dieser Vorgabe dadurch gerecht zu werden, indem er die Ladung in einem sägezahnähnlichen Muster ständig im Bereich von ca. $69 - 71\ \%$ hält, was ihm im ersten Abschnitt der Strecke, wo die elektrische Maschine ständig im motorischen Betrieb arbeitet, auch gelingt. Auf halbem Weg, ca. bei $7000\ m$, wird der lange, steil bergab führende Abschnitt der Strecke erkannt, dadurch lässt der Regler vor diesem Bereich die Ladung auf ca. $66\ \%$ fallen. Der letzte Abschnitt der Strecke ist durch ein sanftes Gefälle charakterisiert, wo abwechselnd mit geringer Leistung angetrieben oder rekuperiert wird. Durch diese Gegebenheit, dem relativ kurzen Prädiktionshorizont und der kontinuierlichen Zielvorgabe von $70\ \%$ kommt der Regler an seine Grenzen: Durch die zu späte Erkennung des vorherigen Rekuperationspotenzials wird die Batterie über diese Zielladung hinaus geladen und durch die geringe Leistungsanforderung und den weiteren Rekuperationsmöglichkeiten nicht mehr belastet. Dadurch wird die Zielladung von $70\ \%$ am Ende der Strecke übertroffen, was eine unnötige Erhöhung der Kosten verursacht.

Abhilfe schafft folgende Idee: Unter der Annahme, dass sich die gewählte Strecke während

der Fahrt nicht ändert und das Steigungsprofil entlang dieser Strecke dadurch gegeben und invariant ist, wird klar, dass die ausschlaggebende Störgröße während der Optimierung die nicht prädizierbaren Änderungen der Fahrzeuggeschwindigkeit sind. Wäre also die Fahrzeuggeschwindigkeit auch konstant gegeben, würde die Berechnung einer optimalen Steuerung wie im Abschnitt 3.3 beschrieben, einen Ladungsverlauf höherer Güte liefern. Ausgehend von dieser Tatsache wird die Referenztrajektorie r_k für den modellprädiktiven Regler folgendermaßen bestimmt: Unter der Annahme einer erwarteten *Durchschnittsgeschwindigkeit* $\bar{v}$ wird mittels dynamischer Programmierung ein prädizierter Ladungsverlauf $\tilde{x}_k$ entlang der gesamten Strecke bestimmt. Dieser Ladungsverlauf liefert zu jedem Endpunkt $\hat{x}_N$ des Prädiktionshorizonts die gewünschte Endladung als $\hat{x}_N = \tilde{x}_k + p$. Dieser Ablauf ist in Abbildung 4.4 dargestellt.

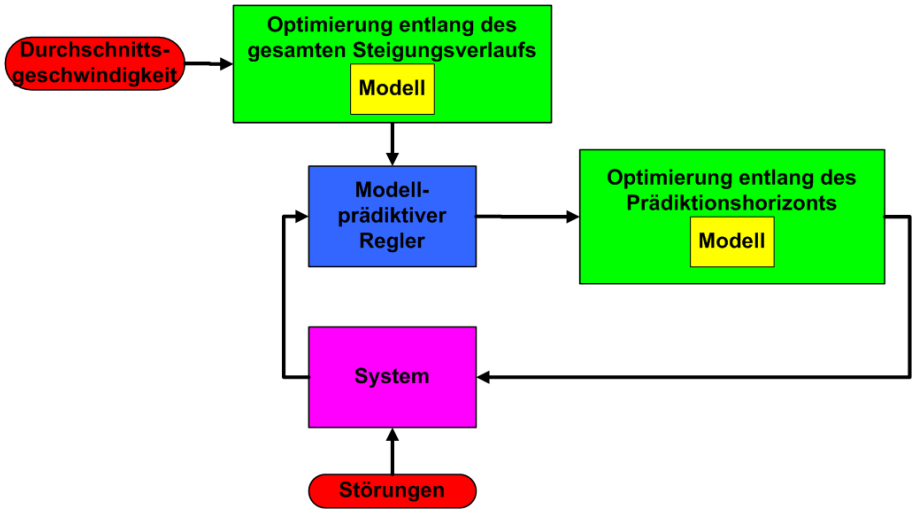


Abbildung 4.4: Modellprädiktives Regelsystem mit Vorberechnung der Referenztrajektorie mittels dynamischer Programmierung

Durch diese Vorgangsweise fließt bei jedem Optimierungsschritt des Reglers die Information über den gesamten Steigungsverlauf in die Berechnung ein. Damit wurde ein großer Anteil der Störungsquellen eliminiert, es verbleiben ausschließlich die Änderungen der Fahrzeuggeschwindigkeit *in Bezug auf* eine prädizierte Durchschnittsgeschwindigkeit und die Ungenauigkeiten des Modells. Ein zusätzlicher positiver Effekt entsteht bei richtiger Wahl der Durchschnittsgeschwindigkeit darin, dass der Regler nurmehr die Differenz zu dieser Durchschnittsgeschwindigkeit ausgleichen muss. Im günstigsten Fall kann die Summe dieser Abweichungen entlang der Strecke sogar Null ergeben.

Ein Vergleich der Ladungsverläufe des modellprädiktiven Reglers mit und ohne Vorprädiktion des Ladungsverlaufs mittels dynamischer Programmierung ist in Abbildung 4.3 dargestellt. Die Vorberechnung liefert eine prädizierte Ladungstrajektorie, die in grün dargestellt ist. Diese Ladungstrajektorie spiegelt den Verlauf des Höhenprofils gut wieder: Durch die Steigung im ersten Abschnitt und dem Gefälle mit hohem Rekuperationspotenzial erreicht der Ladungsverlauf sein Minimum bei $6000\ m$ mit $53\ \%$ und mit der vorberechneten *Steuergröße* wird am Ende der Strecke die Zielladung von $70\ \%$ unter Berücksichtigung aller durch das Höhenprofil gegebene Möglichkeiten und der Aufwendung von minimalen Kosten erreicht.

Nimmt man dieses vorberechnete Ladungsprofil als Referenz für einen modellprädiktiven Regler mit wanderndem kurzem Prädiktionshorizont, werden diese wichtigen Informationen aus dem gesamten Höhenprofil für den Regler zur Verfügung gestellt. Der Verlauf der Ladung mit einem solchen Regler ist in Abbildung 4.3 in blau dargestellt. Der Vergleich mit der Re-

ferenztrajektorie in grün zeigt die Abweichungen, die durch die Verwendung des tatsächlichen Geschwindigkeitsprofils v statt einer konstanen Durchschnittsgeschwindigkeit $\bar{v}$ entstehen. Es ist sofort ersichtlich, dass durch die Wahl dieser Vorgangsweise und der Mitberücksichtigung der Höhenprofilinformationen der Gesamtstrecke der Regler ein besseres Verhalten gegenüber dem Verlauf in rot zeigt und auch im Bezug auf die Kosten ein profitableres Ergebnis liefert.

Dadurch wird die Idee der Vorberechnung einer Referenztrajektorie basierend auf einer angenommenen Durchschnittsgeschwindigkeit entlang der Gesamtstrecke für den modellprädiktiven Regler bestätigt. Wenn man bedenkt, welche Folgen die Vernachlässigung eines langen Gefälles, nämlich die Überladung der Batterie über die zulässigen Betriebsgrenzen, haben kann und welches Potenzial in dieser Information für die gesamte Betriebsstrategie enthalten ist, wird für viele Strecken die Anwendung dieser Vorgangsweise womöglich der einzing gangbare Weg sein, um ein zufriedenstellendes Ergebnis zu erhalten.

Kapitel 5

Evaluierung der modellprädiktiven Regelung

In diesem Kapitel werden die Ergebnisse der modellprädiktiven Regelung aus Kapitel 4 für die im Abschnitt 2.4 vorgestellten Fahrstrecken vorgestellt. Damit nicht nur die statischen Verläufe nach erfolgter Regelung dargestellt werden können, wurde ein MATLAB®-Skript implementiert (Abschnitt 5.1), mit dessen Hilfe die Fahrt entlang der gewählten Strecke dynamisch in einer aus Google Maps stammenden Karte dargestellt wird.

Die tatsächliche Diskussion der Ergebnisse erfolgt im Abschnitt 5.2. Dabei werden sowohl Parametervariationen des Modells durch Einbindung eines AVL Cruise®-Modells als auch Regelszenarien, wo der Fahrer die gewählte Strecke während der Fahrt ändert, betrachtet. Anschließend wird der Einfluss der Geschwindigkeit auf die Kosten untersucht.

5.1 Visualisierung der Ergebnisse

Die Visualisierung beinhaltet drei Teilbilder (siehe Abbildung 5.1): Die Karte mit dem nach Betriebszustand eingefärbten Streckenverlauf, das Höhenprofil und schließlich eine Systemübersicht des Fahrzeugs, die den Energiefluss im aktuellen Betriebszustand darstellt (Abb. 5.2)[1].

Für die Karte gilt folgende Legende:

Schwarz-weiss: Noch nicht abgefahrene Punkte der prädizierten Strecke

Blau: Streckenabschnitte, wo die elektrische Maschine im motorischen Betrieb arbeitet und der Range Extender ausgeschaltet ist (Abb. 5.2a)

Grün: Streckenabschnitte, wo die elektrische Maschine rekuperiert und der Range Extender ausgeschaltet ist (Abb. 5.2b)

Rot: Streckenabschnitte, wo die elektrische Maschine im motorischen Betrieb arbeitet und der Range Extender eingeschaltet ist (Abb. 5.2c)

[1] Die Komponenten Verbrennungskraftmaschine, Rad, elektrische Maschine und die Batterie in dieser Abbildung stammen aus der frei verfügbaren Datenbank von Google SketchUp, an dieser Stelle Dank an die Benutzer *Mo*, *(((ILKE®)))*, *COL 1* und *André da Silva* für die Bereitstellung ihrer Arbeit.

Gelb: Streckenabschnitte, wo die elektrische Maschine rekuperiert und der Range Extender eingeschaltet ist (Abb. 5.2d)

Dünn strichlierte Farben: Streckenpunkte, die vom modellprädiktiven Regler als Teil des Prädiktionshorizonts vorberechnet worden sind

Während der Visualisierung werden die äquidistant entlang der Strecke verteilten Ergebnisse aus der modellprädiktiven Regelung auf die Stützstellen der GPX-Daten, die im Allgemeinen nicht im selben Abstand verteilt sind, interpoliert und der Betriebszustand zu jedem Punkt ausgewertet. Durch diese Umrechnung kommt es zu Ungenauigkeiten bei der Darstellung und der Anzeige der Leistungs- und Ladungswerte in der Betriebszustandsübersicht, die jedoch für die Visualisierung hinnehmbar aber nicht für quantitative Aussagen geeignet sind.

Das Laden der Google Maps Karte für die Visualisierung in MATLAB® wird mit Hilfe der Funktion `plot_google_map.m` realisiert, die bis auf einige kleiner Modifikationen vom MATLAB® Central Benutzer *Zohar Bar-Yehuda* zur Verfügung gestellt worden ist[2]. Die Funktion lädt zu den in geographischen Koordinaten gegebenen Streckenpunkten passenden Kartenausschnitt vom Google Maps Server und stellt diese nach Umrechnung des pixel-basierten Bildes auf geographische Koordinaten in einem MATLAB®-Figure dar. Weiters wird ein Zoomen und Verschieben dieser Karte unterstützt, wofür die aktualisierte Karte neu geladen wird. Jedoch ist diese Funktion sparsam zu verwenden, da Google Maps die maximale Anzahl an täglichen Zugriffen durch diese Funktion auf 1000 limitiert.

[2] `http://www.mathworks.com/matlabcentral/fileexchange/27627-plotgooglemap`

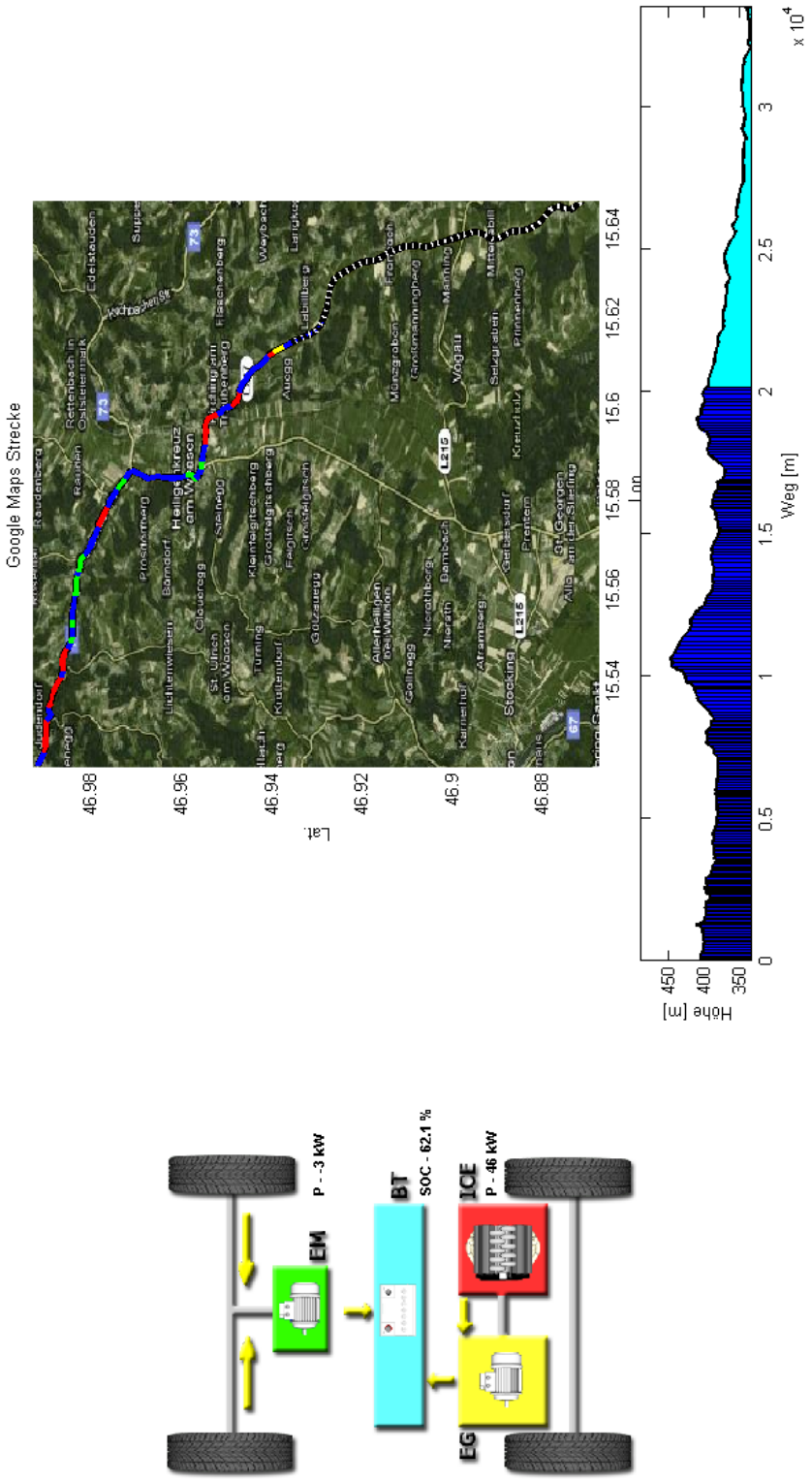
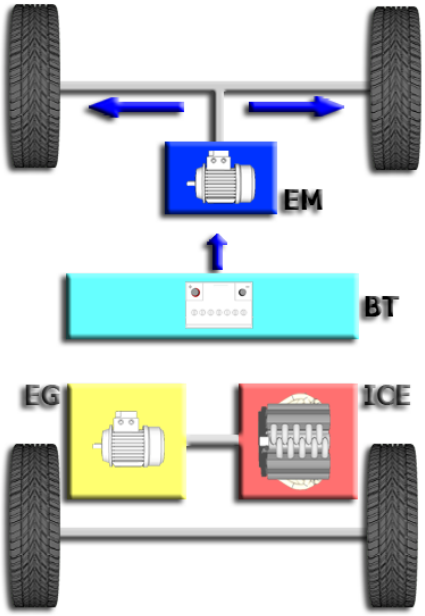
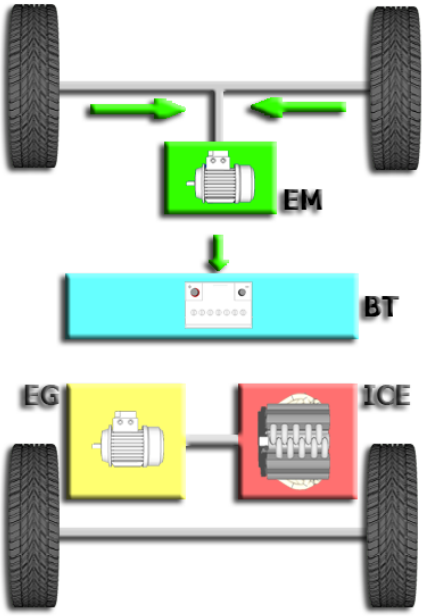


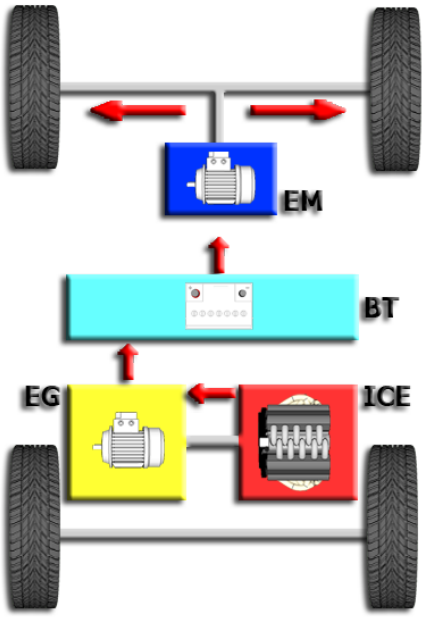
Abbildung 5.1: Oberfläche für die Visualisierung der Ergebnisse



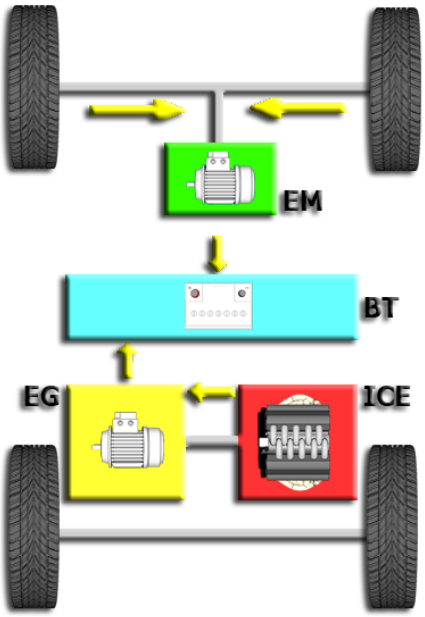
(a) EM motorisch



(b) EM generatorisch



(c) EM motorisch mit Range Extender



(d) EM generatorisch mit Range Extender

Abbildung 5.2: Darstellung der Betriebszustände eines Serienhybrids

5.2 Diskussion der Evaluierungsergebnisse

5.2.1 Evaluierung der Rechberg-Strecke mit AVL Cruise®

Die Anwendung der modellprädiktiven Regelung auf die Rechberg-Strecke mit einem Prädiktionshorizont von $1\ km$ ist in Abbildung 5.5 dargestellt und mit Hilfe der auf der beiliegenden CD-ROM gespeicherten Videodatei `RechbergStrecke.avi` visualisiert. Wie erwartet, erfüllt der Regler die Zielvorgabe von 70% Endladung.

Um die Robustheit des Reglers gegen mögliche Paramtervariationen des Modells abzuschätzen, wurde in Simulink ein Modell erstellt, wo der aktuelle Ladungszustand x_k der Batterie nicht mit Hilfe der quasistatischen Modellbildung mit MATLAB® berechnet wird, sondern über eine Schnittstelle vom AVL Cruise®-Modell *Range Extender* geliefert wird. Diese Schnittstelle überträgt den vom modellprädiktiven Regler bestimmten Schaltzustand $\mu_{0,k}$ der Verbrennungsmaschine an das AVL Cruise®-Modell und überträgt den damit berechneten, aktuellen Ladungszustand $\hat{x}_0$ der Batterie als Anfangszustand für den nächsten Optimierungsdurchlauf an den modellprädiktiven Regler. Diese Schnittstelle ist in Abbildung 5.3 dargestellt.

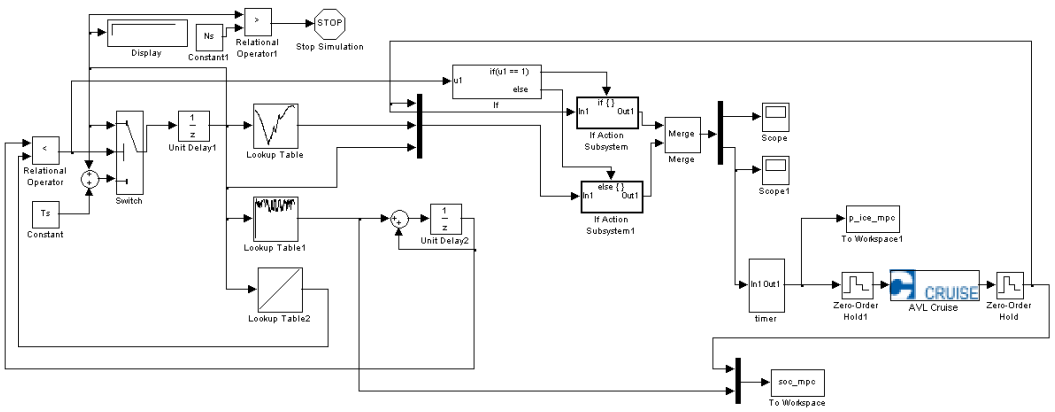


Abbildung 5.3: Simulink-Modell zur Einbindung einer AVL Cruise®-Schnittstelle während der modellprädiktiven Regelung

Der Vergleich der Regelverläufe zwischen dem quasistatischen Modell und dem AVL Cruise®-Modell ist aus Abbildung 5.4 ersichtlich. Erwartungsgemäß wirken sich die Unterschide der beiden Modelle auf den Regelverlauf aus. Besonders im Ladebetrieb weist das AVL Cruise®-Modell ein anderes Verhalten auf, dieses wird aber durch den Regler bis zum Erreichen der Zielvorgabe ausgeglichen. Diese Änderung des Modells hat auch eine Auswirkung auf die Kosten für das Erreichen der Zielvorgabe.

Erwartungsgemäß sind die Kosten bei einer optimalen Steuerung am niedrigsten: Hier wird davon ausgegangen, dass alle Einflüsse, die entlang der gesamten Strecke auf das Fahrzeug wirken werden, im Voraus bekannt sind und daher für die Bestimmung einer optimalen Steuerfolge herangezogen werden können. Nachteil dieser Lösung ist aber die Anfälligkeit auf jede Störung, die auftritt. Bei der modellprädiktiven Regelung ohne Parametervariation wird angenommen, dass das Fahrzeug ausreichend genau modelliert ist, sodass vom Modell keine Störungen während der Fahrt ausgehen. Als einzige Störquelle muss vom Regler nur eine Änderung im Geschwindigkeitsverlauf berücksichtigt werden, wobei im Allgemeinen dieser Vorgangsweise in der Realität auch Grenzen gesetzt sind. Wird der aktuelle Zustand wie im

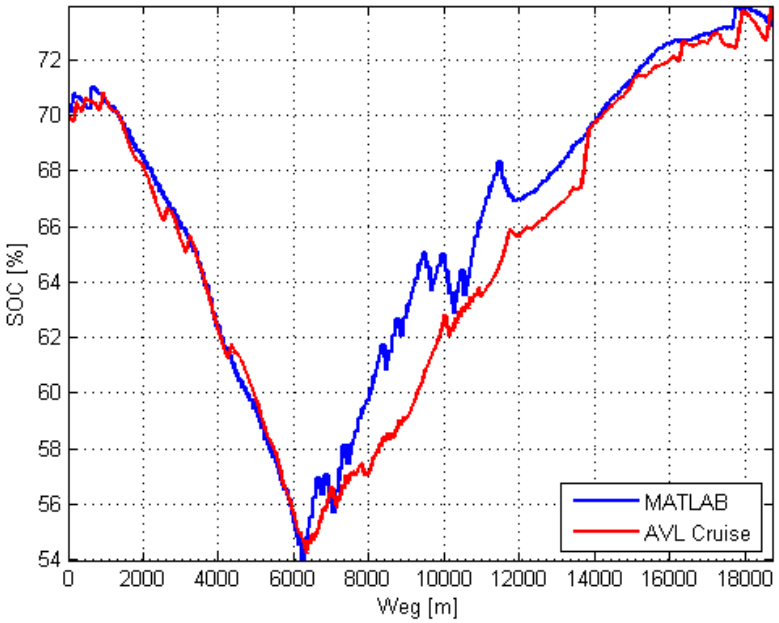


Abbildung 5.4: Vergleich der Regelverläufe bei Einbindung des AVL Cruise®-Modells

dritten Fall von einem externen Modell - oder in der Realität als gemessener Wert im Fahrzeug - für die modellprädiktive Regelung herangezogen, muss diese auch Störungen, die durch eine Modellungenauigkeit durch Paramatervariationen zustande kommen, entgegenwirken. Dabei ist nicht schon im Voraus klar, ob diese eine Erhöhung oder eine Verminderung der Kosten bewirken, beide Fälle wären in der Realität möglich und für das Erreichen der Zielvorgabe relevant.

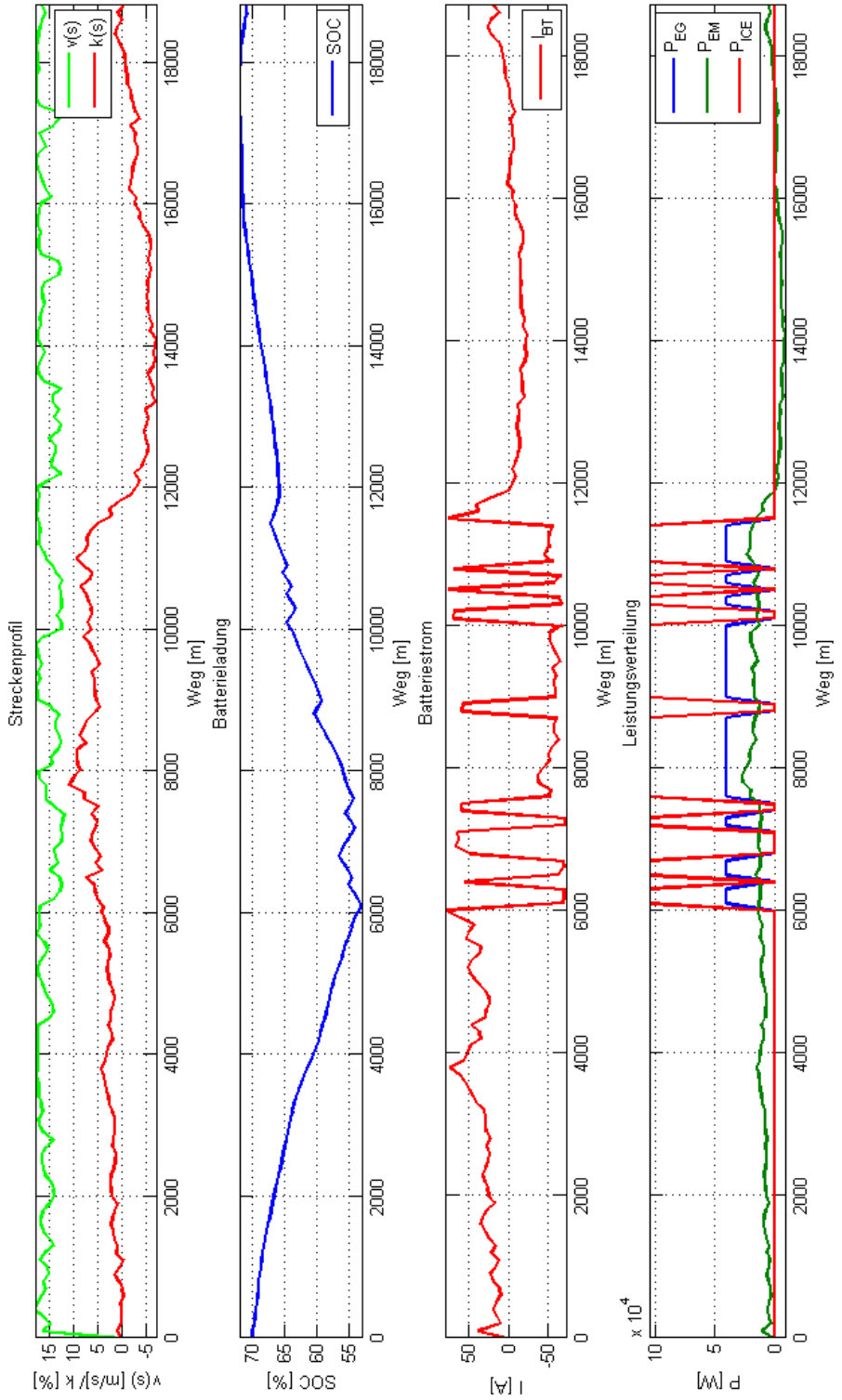


Abbildung 5.5: Verlauf der wichtigsten Fahrzeuggrößen während der modellprädiktiven Regelung an der Rechberg-Strecke

5.2.2 Streckenänderung während der modellprädiktiven Regelung

Ein weiteres wichtiges Szenario ist die Änderung der Strecke während der Fahrt, die durch einen entsprechenden Fahrerwunsch oder durch Änderung der Verkehrssituation durch Staus oder Umleitungen jederzeit möglich ist. Tritt dieses Szenario auf, muss der Regler unverzüglich darauf reagieren, um überhaupt noch eine Chance zur Erfüllung der Zielvorgabe entlang der neuen, geänderten Strecke erfüllen zu können. Unter der Annahme, dass der Regler durch eine entsprechende Umgebung (Navigationssystem mit Kartenmaterial) über eine Änderung der Strecke Informationen erhält und die Daten der geänderten Streckenführung zur Verfügung gestellt bekommt, erfolgt die Reaktion des Reglers nach folgenden Punkten:

1. Streckenänderung wird erkannt
2. Aktueller Optimierungsschritt wird abgebrochen, Prädiktionshorizont nicht mehr aktuell, Ergebnisse müssen verworfen werden
3. Bestimmung des aktuellen Ladungszustands x_k
4. Bestimmung der neuen Referenztrajektorie für die geänderte Strecke
5. Neuer Reglerablauf mit dem neuen Anfangzustand $\bar{x}_0 = x_k$ entsprechend dem aktuellen Ladungszustand entlang der geänderten Strecke mit dem neuen Prädiktionshorizont

Damit ein solches Szenario nachgestellt werden kann, wurden bei den Strecken Budapest und Graz-Wolfsberg Umleitungsschleifen aufgenommen. Dabei sind die Anfangs- und Endpunkte der Strecken gleich, es wurde entlang der Strecken eine Umleitung eingefügt, wodurch die Gesamtstreckenlänge größer geworden ist und sich das Steigungsprofil geändert hat. Entsprechend dem vorhin geschilderten Szenario wurde bei Bekanntwerden der Streckenänderung - typisch 5-10 Schritte vor der Verzweigung - der Optimierungsablauf des aktuellen Prädiktionshorizonts abgebrochen und ein neuer Ablauf mit dem aktuellen Ladungszustand und der geänderten Strecke gestartet. Die damit erhaltenen Verläufe sind den Abbildungen 5.6-5.7 zu entnehmen, dabei repräsentiert der vertikale schwarze Strich den Zeitpunkt der Erkennung der Umleitung[3].

Wie aus den Verläufen 5.6-5.7 ersichtlich, besitzt der modellprädiktive Regler die Robustheit, um mit diesen Umleitungszenarien umzugehen und erfüllt die Zielvorgabe auch unter einer geänderten Streckenführung. An dieser Stelle ist aber anzumerken, dass es im Allgemeinen keine Garantie dafür gibt, dass jede mögliche Umleitung zu jedem möglichen Zustand zu einer gültigen Zielladung führt. Es muss zumindest gewährleistet werden, dass in der Zeit, die für das Abfahren der geänderten Reststrecke zur Verfügung steht, die Zielladung bei durchgehendem Betrieb des Range Extenders erreicht werden kann. Weiters ist davon auszugehen, dass die Kosten für das Abfahren einer Streckenführung, die sich während der Regelung geändert hat, im Allgemeinen höher sein wird, als im Fall, dass die gesamte Strecke vom Anfang an bekannt gewesen wäre.

[3] Visualisierungen auf beiliegender CD-ROM

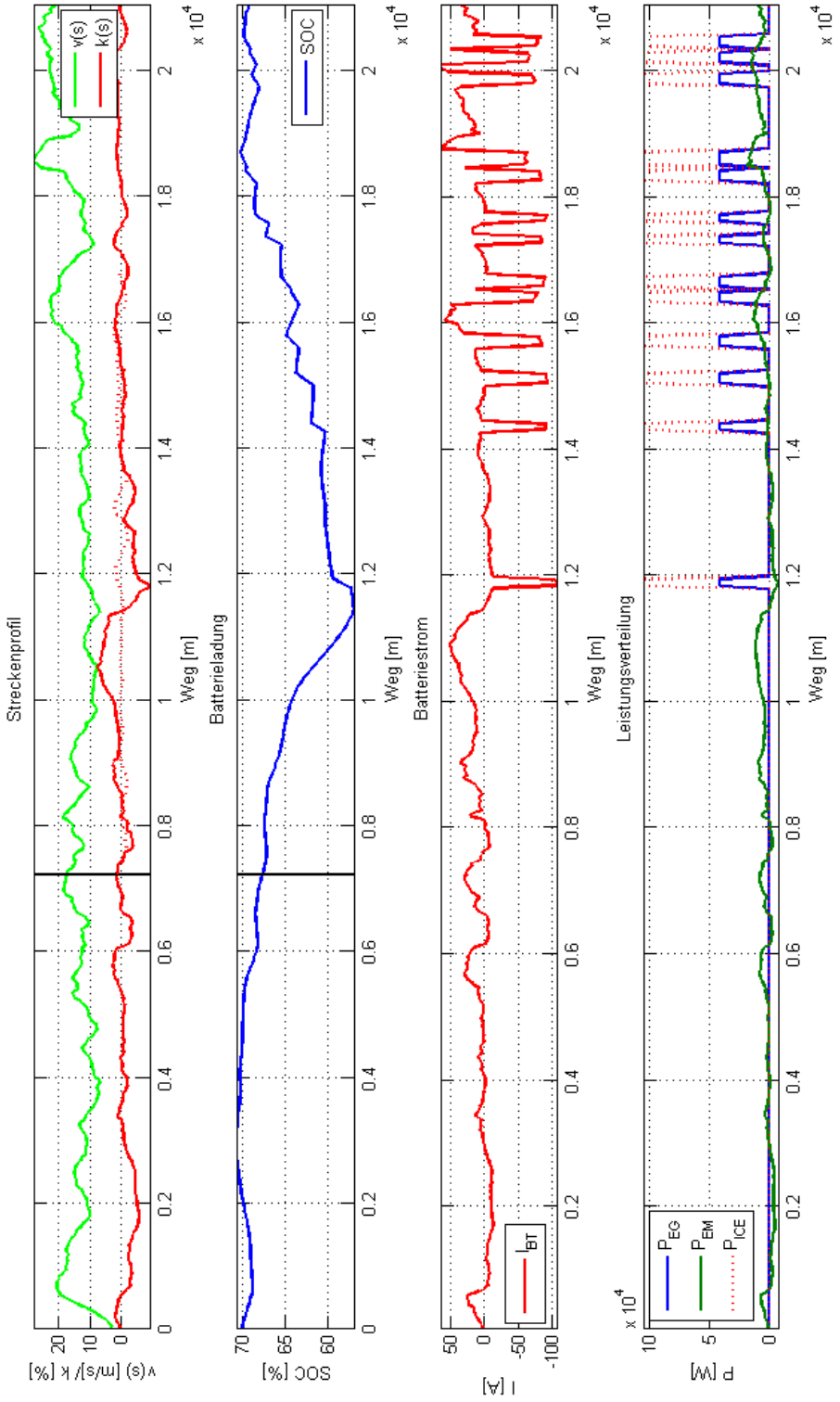


Abbildung 5.6: Verlauf der modellprädiktiven Regelung entlang der Stadtstrecke Budapest mit Streckenänderung

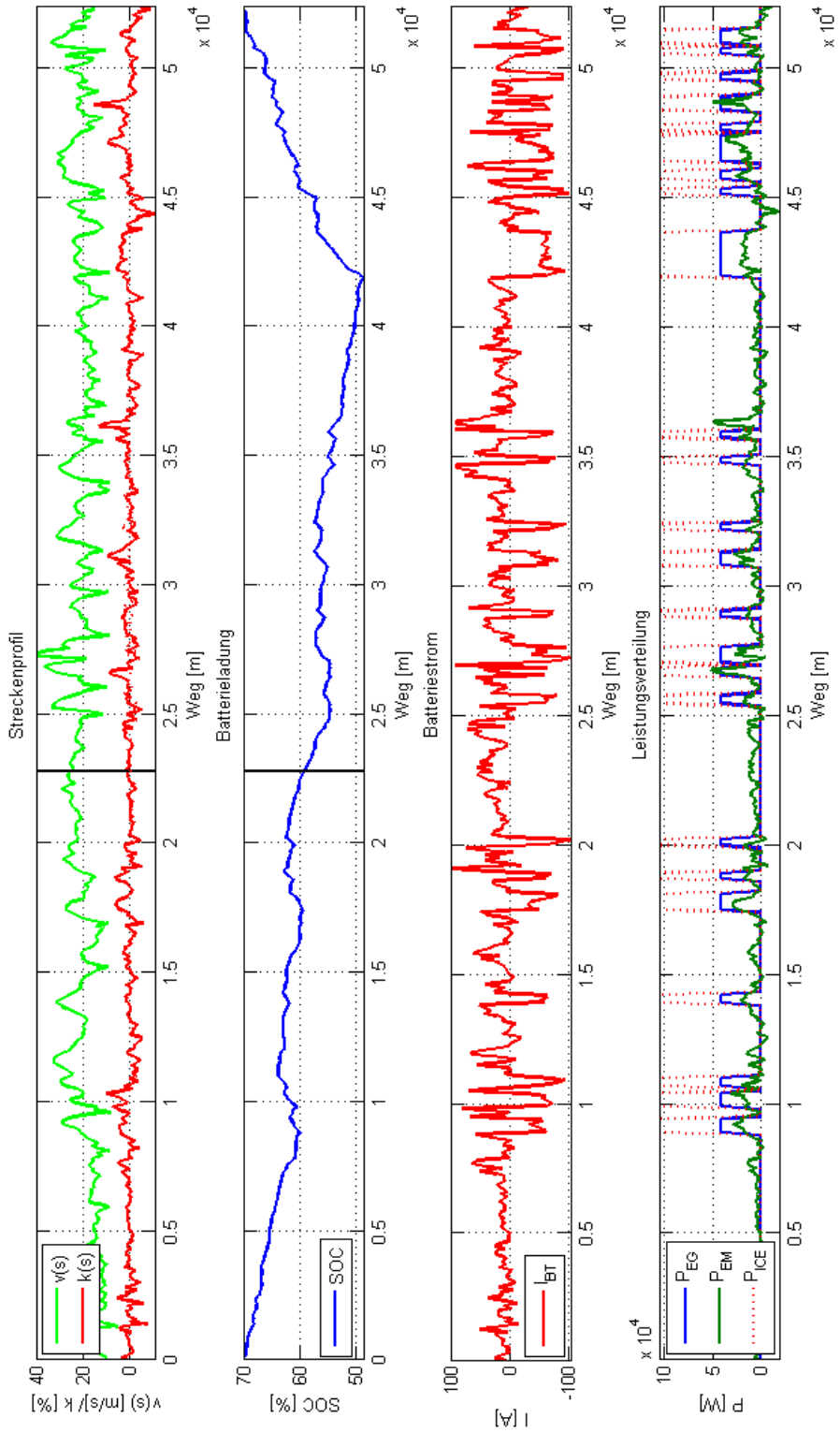


Abbildung 5.7: Verlauf der modellprädiktiven Regelung entlang der Graz-Wolfsberg-Strecke mit Streckenänderung

5.2.3 Einfluss der Geschwindigkeit auf die Kosten

Durch die Verwendung der modellprädiktiven Regelung bietet sich eine weitere Funktion dieser Vorgangsweise an: Da der Regler kontinuierlich den Einfluss der vorgegebenen Fahrgeschwindigkeit auf die Kosten evaluiert, könnte die Funktion des Reglers erweitert werden, indem dieser dem Fahrer eine Rückmeldung über das Kraftstoffeinsparpotenzial durch die Wahl einer entsprechenden Geschwindigkeit im aktuellen Streckenabschnitt gibt. Diese Funktion wurde zwar in dieser Arbeit nicht implementiert, jedoch sind in Tabelle 5.1 verschiedene Kosten der Graz-Wolfsberg-Strecke bei verschiedenen Obergrenzen und Durchschnittsgeschwindigkeiten aufgelistet.

Durchschnitts-geschwindigkeit $\left[\frac{km}{h}\right]$	**Maximal-geschwindigkeit** $\left[\frac{km}{h}\right]$	**Rekuperierte Energie** $[KWh]$	**Kosten** $\left[\frac{l}{100km}\right]$	**Bemerkung**
72.58	145.55	0.381	3.467	Originalstrecke
65.32	131	0.422	3.372	Geschwindigkeit um 10% entlang der Strecke gemindert
71.56	100.8	0.3866	3.405	Geschwindigkeit auf $100.8\frac{km}{h}$ beschränkt
66.93	79.2	0.426	3.164	Geschwindigkeit auf $79.2\frac{km}{h}$ beschränkt

Tabelle 5.1: Kostenvergleich zwischen optimaler Steuerung und modellprädiktivem Regler mit und ohne Parametervariation.

Anhand Tabelle 5.1 ist ersichtlich, dass die Wahl der Geschwindigkeit - vor allem der maximalen Geschwindigkeit - einen entscheidenden Einfluss auf die Kosten hat. Interessant ist auch die Aufstellung über das Potential der Rekuperierung in Abhängigkeit der Geschwindigkeit. Denn je höher die Geschwindigkeitsvorgabe an einem Gefälle ist, desto weniger Bremsenergie steht zur Rekuperation zur Verfügung. Daraus folgt, dass die Erweiterung der Funktion des Reglers für die Bestimmung eines optimalen Geschwindigkeitsprofils unter Einhaltung einer Zeitbegrenzung für die Gesamtfahrzeit sinnvoll ist. Jedoch muss dabei beachtet werden, dass die Bewertung einer Zeitbegrenzung zu einer Erweiterung der Optimierung auf ein System zweiter Ordnung führt, womit die Rechenzeit erheblich gesteigert und für eine Berechnung während der Fahrzeit unter Umständen ungeeignet wird.

Kapitel 6

Zusammenfassung und Ausblick

In dieser Arbeit wurde gezeigt, dass mit Hilfe der modellprädiktiven Regelung (Kapitel 4), die sich auf die Anwendung der dynamischen Programmierung als Optimierungsverfahren stützt (Kapitel 3), ein leistungsfähiges Werkzeug für die Bestimmung einer optimalen Betriebsstrategie für ein Serienhybridfahrzeug gegeben ist, das sogar während der Fahrzeit entlang einer realen Strecke eingesetzt werden könnte.

Voraussetzung für eine funktionsfähige Implementierung ist das Vorhandensein eines entsprechenden Kartenmaterials, das die Strecke genau genug abbildet. Weiters wurde stets die Annahme getroffen, dass der Fahrer sein Fahrziel bekannt gibt und nicht ändert - obwohl eine Änderung des Fahrziels ähnlich zu einer Streckenänderung behandelt werden könnte. Mit Hilfe heutiger Navigationstechnik stellt diese Forderung aber keine Schwierigkeit dar, es existieren sogar schon Hybridfahrzeuge[1], die bereits solche Daten für ihren Betrieb berücksichtigen.

Die Robustheit eines modellprädiktiven Reglers ist entscheidend für die Beurteilung seiner Güte, dafür wurden während der Evaluierung im Kapitel 5 Einflüsse von Parametervariationen im Modell, sowie Änderungen der Strecke auf das Regelverhalten überprüft. Dabei hat der Regler stets die Zielvorgabe erfüllt.

Im Abschnitt 5.2.3 wurde untersucht, welchen Einfluss die Geschwindigkeit auf die Kosten hat. Für zukünftige Arbeiten wäre es möglich, die Funktion des modellprädiktiven Reglers dahingehend zu erweitern, dass dieser dem Fahrer unter Einhaltung gewisser Zeitgrenzen für die Gesamtstrecke eine Empfehlung über das kostengünstige Geschwindigkeitsprofil gibt.

In diesem Zusammenhang könnte in der Zukunft das Regelverhalten verbessert werden, indem eine streckenabhängige Geschwindigkeitsprädizierung vorgenommen wird. Durch die bekannte Streckenführung kann man Rückschlüsse auf die Geschwindigkeit in Kurvenfahrten, Kreuzungen und Geschwindigkeitsbegrenzungen ziehen. Weiters könnte der Regler den Fahrstil des Fahrers erlernen, woraus eine noch genauere Prädizierung der Geschwindigkeit und damit verbunden des Ladungszustands möglich wird.

[1] BMW Concept 5er ActiveHybrid `http://www.bmw.de/de/de/insights/technology/efficientdynamics/phase_1/5series_activehybrid_energy_management.html`

Anhang A

CD-ROM

Auf beiliegender CD-ROM sind folgende Videodateien zur Visualisierung der Ergebnisse aus Kapitel 5 zu finden:

- Rechberg-Strecke: `RechbergStrecke.avi`
- Budapest-Strecke: `BudapestStrecke.avi`
- Graz-Wolfsberg-Strecke: `GrazWolfsbergStrecke.avi`

Weiters enthält die CD-ROM diese Arbeit als PDF-Datei.

Anhang B

MATLAB®-Skripts

B.1 Translatorische Beschreibung des Fahrzeugs

```
%% Vehicle Modell Function
%  Inputs: speed, acceleration, input parameters
%  Outputs: rotatory wheel speed, rotatory wheel acceleration,
   torque
function hdmf_vhcl = hdmf_vehicle_modell(v_of_s,dv_of_s,incl_vec,
   inpars)

%% Input parameters:
%  r_wheel: wheel radius
%  m_f: vehicle mass
%  Af: vehicle surface
%  cw: CW-value
%  rtm: rotatory mass percentage


% Constants
g = 9.81;
ur = 0.008;
v_tol = 0.1;
rho = 1.18;

% Transitional/rotatory conversion
w_wheel = v_of_s./inpars.r_wheel;
dw_wheel = dv_of_s./inpars.r_wheel;

%% Losses
% Rolling friction
F_roll = inpars.mf*g*ur.*gt(abs(v_of_s),v_tol);
% Aerodynamic losses
F_aero = 0.5*rho*inpars.Af*inpars.cw.*v_of_s.^2;
% Additional force due to road incline
F_incl = inpars.mf*g*sin(incl_vec);
```

```
31 % Inertia
32 J = (inpars.mf*(1+inpars.rtm/100)*inpars.r_wheel^2)/inpars.ds.*
       dw_wheel;
33
34 %% Torque calculation
35 M = (F_roll+F_aero+F_incl).*inpars.r_wheel+J;
36
37 %% Output structure
38 hdmf_vhcl = struct('w_wheel',w_wheel,'dw_wheel',dw_wheel,'M_wheel',
       M);
```

B.2 Das Getriebe

```
1  %% Transmission modell function
2  %  Inputs: rot. speed, rot. acceleration, torque, input parameters
3  %  Outputs: geared rot. speed, rot. acceleration and torque
4  function hdmf_trans = hdmf_transmission(w_in,dw_in,M_in,inpars)
5  %% Input parameters:
6  %  gr: gear ratio
7  %  P_idle: gearbox idling power
8  %  etr: transmission efficiency constant
9
10 %% Constants
11 zero_tol = 0.01;    % Zero deviation tolerance
12 inf_val = 10^6;     % "Infinite" value
13
14 %% Calculating geared speed and acceleration
15 w_out = w_in*inpars.gr;
16 dw_out = dw_in*inpars.gr;
17
18 % P_idle = inpars.P_idle*gt(abs(w_in),inpars.w_min).*sign(w_in.*
       M_in);
19 P_idle = 0;
20
21 %% Calculating geared torque
22 %  Division by 0 prevention
23 div0 = (abs(w_out) < zero_tol)*inf_val;
24 M_out = (inpars.etr*M_in.*w_in-P_idle)./(w_out+div0);
25
26 %% Output structure
27 hdmf_trans = struct('w_out',w_out,'dw_out',dw_out,'M_out',M_out);
```

B.3 Die elektrische Maschine

```
%% Electic motor modell function
%  Inputs: rot. speed, rot. acceleration, torque, input parameters
%  Outputs: electric motor power, overspeed & overload error
function hdmf_EM_out = hdmf_EM(w_in,dw_in,M_in,inpars)

%% Input parameters:
%  scale_EM: electric motor scaling factor
%  J_EM: electric motor inner inertia
%  P_aux: power of auxiliary devices

%% Load electic motor map
load eta_EM_map                         % Efficiency map
                  [-]

load w_EM_row                           % Motor speed range
                [rad/s]
load T_EM_col                           % Motor torque range
               [Nm]
load w_EM_max                           % Maximum motor speed
             [rad/s]
load T_EM_max                           % Maximum motor torque
            [Nm]

w_EM_row = w_EM_row*1.75;
w_EM_max = w_EM_max*1.75;

w_EM_upper = max(w_EM_max);             % Upper limit motor speed
         [rad/s]


%% Scale motor torque
M_EM_col = inpars.scale_EM * T_EM_col;
M_EM_max = inpars.scale_EM * T_EM_max;

%% Calculate torque
M_EM = M_in+inpars.J_EM*dw_in;

%% Lookup efficiency value
[M_EM_col_mesh,w_EM_row_mesh] = meshgrid(M_EM_col,w_EM_row);
eta_EM=interp2(M_EM_col_mesh,w_EM_row_mesh,eta_EM_map,M_EM,w_in,'
   linear',0);

error = 0;
inf_out = size(length(w_in));           % Infeasibility array

%% Check for overspeed violation
inf_out = (w_in > w_EM_upper);
[max_w_in,i_max_w_in] = max(w_in);
```

```
42 if(max_w_in > w_EM_upper)
43     disp(sprintf('Error: w_EM_max = %f exceeded at point %d.\n',
          w_EM_upper,...
44               i_max_w_in))
45     error = 1;
46 end
47
48 %% Check for overload violation
49 inf_out = inf_out+(interp1(w_EM_row',M_EM_max,w_in,'linear',0)<abs(
       M_EM));
50 [min_M,i_min_M]=min(interp1(w_EM_row',M_EM_max,w_in,'linear',0)-abs
       (M_EM));
51 if(min_M < 0)
52     disp(sprintf('Error: M_EM_max = %f exceeded at point %d.\n',
          M_EM_max,...
53           i_min_M))
54     error = error + 2;
55 end
56
57 %% Calculate output power
58 P_EM = inpars.P_aux+M_EM.*eta_EM.*w_in;
59
60 %% Output structure
61 hdmf_EM_out = struct('P_EM',P_EM,'error_EM',error,'Inf_EM',inf_out)
       ;
```

B.4 Der elektrische Generator

```
1
2 %% Electic generator modell function
3 %  Inputs: rot. speed, rot. acceleration, torque, input parameters
4 %  Outputs: electric generator power, overspeed & overload error
5 function hdmf_EG_out = hdmf_EG(w_in,M_in,inpars)
6
7 %% Input parameters:
8 %  scale_EG: electric generator scaling factor
9
10 %% Load electic generator map
11 load eta_EG_map                    % Efficiency map
                  [-]
12
13 load w_EG_row                      % Generator speed range
            [rad/s]
14 load T_EG_col                      % Generator torque range
           [Nm]
15 load w_EG_max                      % Maximum generator speed
          [rad/s]
```

```
16 load T_EG_max                          % Maximum generator torque
       [Nm]
17
18 w_EG_row = w_EG_row*1.75;
19 w_EG_max = w_EG_max*1.75;
20
21 w_EG_upper = max(w_EG_max);          % Upper limit generator speed
     [rad/s]
22
23
24 %% Scale motor torque
25 M_EG_col = inpars.scale_EG * T_EG_col;
26 M_EG_max = inpars.scale_EG * T_EG_max;
27
28 %% Lookup efficiency value
29 [M_EG_col_mesh,w_EG_row_mesh] = meshgrid(M_EG_col,w_EG_row);
30 eta_EG = interp2(M_EG_col_mesh,w_EG_row_mesh,eta_EG_map,M_in,w_in
     ,...
31                 'linear',0);
32
33 error = 0;
34 inf_out = zeros(size(w_in));          % Infeasibility array
35
36 %% Check for overspeed violation
37 inf_out = (w_in > w_EG_upper);
38 [max_w_in,i_max_w_in] = max(w_in);
39 if(max_w_in > w_EG_upper)
40     disp(sprintf('Error: w_EG_max = %f exceeded at point %d.\n',
         w_EG_upper,...
41             i_max_w_in))
42     error = 1;
43 end
44
45 %% Check for overload violation
46 inf_out = inf_out+(interp1(w_EG_row',M_EG_max,w_in,'linear',0)<abs(
     M_in));
47 [min_M,i_min_M]=min(interp1(w_EG_row',M_EG_max,w_in,'linear',0)-abs
     (M_in));
48 if(min_M < 0)
49     disp(sprintf('Error: M_EG_max = %f exceeded at point %d.\n',...
50          M_EG_max,i_min_M))
51     error = error + 2;
52 end
53
54 %% Calculate output power
55 P_EG = M_in.*eta_EG.*w_in;
56
57 %% Output structure
58 hdmf_EG_out = struct('P_EG',P_EG,'error_EG',error,'Inf_EG',inf_out)
     ;
```

B.5 Die Verbrennungskraftmaschine

```
%% Internal combustion engine modell function
%  Inputs: rot. speed, rot. acceleration, torque, input parameters
%  Outputs: internal combustion engine power, overspeed & overload
    error
function hdmf_ICE_out = hdmf_ICE(w_in,dw_in,M_in,inpars)

%% Input parameters:
%  engine_type: otto or diesel
%  ICE_disp: Displacement
%  scale_ICE: ICE scaling factor
%  J_ICE: engine inertia
%  w_idle: engine speed at idle
%  P_idle: engine power at idle
%  P_aux: power for auxiliary devices
%  M_cutoff: engine torque at fuel cutoff
%  P_cutoff: engine power at fuel cutoff

%% Load ICE map
load V_CE_map          % Consumption map
    [kg/s]
load eta_CE_map        % Efficiency map
    [-]
load w_CE_row          % Engine speed range
    [rad/s]
load p_me_col          % Brake mean effective pressure range
    [Pa]
load w_CE_max          % Maximum engine speed
    [rad/s]
load p_me_max          % Maximum brake mean effective pressure
    [Pa]

w_ICE_upper = max(w_CE_max);               % Max engine speed  [rad
    /s]

% Scale consumption map
V_d      = inpars.ICE_disp/1000;         % Displaced volume  [l] ->
     [m^3]
V_CE_map   = V_CE_map .* inpars.scale_ICE; % Scale engine
    consumption

% Calculate torque values
M_ICE_col   = p_me_col .* V_d/(4*pi);       % Torque range      [Nm
    ]
M_ICE_max   = p_me_max .* V_d/(4*pi);       % Maximum torque    [Nm
    ]

M_ICE_idle  = inpars.P_idle/inpars.w_idle;  % Torque at idle

```

```
% Define engine type
switch inpars.engine_type

    case 1                                          % -> Otto
        H_u         = 42.7e6;                       % Bosch-Manual
       %rho_f       = 0.745;                        % Bosch-Manual

    case 2                                          % -> Diesel
        H_u         = 42.5e6;                       % Bosch-Manual
       %rho_f       = 0.84;                         % Bosch-Manual
end

%% Calculate torque
M_ICE = M_in+inpars.J_ICE*dw_in;
%  Detect fuel cutoff
det_fc = (M_ICE < inpars.M_cutoff);
%  Apply fuel cutoff torque for fuel cutoff operation
M_ICE = M_ICE.*(~det_fc)+inpars.M_cutoff*det_fc;

%% Apply idle rotatory speed
det_w_id = (w_in < inpars.w_idle);
w_in = w_in.*(~det_w_id)+inpars.w_idle*det_w_id;

%% Lookup efficiency value
[M_ICE_col_mesh,w_ICE_row_mesh] = meshgrid(M_ICE_col,w_CE_row);
Vd = interp2(M_ICE_col_mesh,w_ICE_row_mesh,V_CE_map,M_ICE,w_in,'
   linear',0);

error = 0;
inf_out = zeros(length(w_in));

%% Check for overspeed violation
inf_out = (w_in > w_ICE_upper);
[max_w_in,i_max_w_in] = max(w_in);
if(max_w_in > w_ICE_upper)
    fprintf('Error: w_ICE_max = %f exceeded at point %d.\n',...
         w_ICE_upper,i_max_w_in)
    error = 1;
end

%% Check for overload violation
inf_out=inf_out+(interp1(w_CE_row',M_ICE_max,w_in,'linear',0)>abs(
   M_ICE));
[min_M,i_min_M] = min(interp1(w_CE_row',M_ICE_max,w_in,'linear',0)
   ...
                  -abs(M_ICE));
if(min_M < 0)
    fprintf('Error: M_ICE_max = %f exceeded at point %d.\n',
       M_ICE_max,...
         i_min_M)
```

```
83      error = error + 2;
84  end
85
86  %% Calculate output power
87  P_ICE = Vd*H_u;
88  %  Apply idle power
89  det_p_idle = (w_in <= inpars.w_idle).*(M_ICE <= M_ICE_idle);
90  P_ICE = P_ICE.*(~det_p_idle)+inpars.P_idle*det_p_idle;
91  %  Apply cutoff power
92  det_p_cutoff = (M_ICE <= inpars.M_cutoff);
93  P_ICE = P_ICE.*(~det_p_cutoff)+inpars.P_cutoff*det_p_cutoff;
94
95  %% Output structure
96  hdmf_ICE_out = struct('P_ICE',P_ICE,'error_ICE',error,'Inf_ICE',
        inf_out);
```

B.6 Die Batterie

```
1   %% Battery modell function
2   %  Inputs: battery power, input parameters
3   %  Outputs: state of charge change, overload error
4   function hdmf_BT_out = hdmf_BT(P_in,SOC_in,ds,inpars)
5
6
7   %% New version
8   %% Input parameters
9   %  SOC_list: lookup table for SOC values
10  %  U_ocv: lookup table for open-circuit voltage
11  %  R_dis: lookup table for discharging resistance
12  %  R_chg: lookup table for charging resistance
13  %  e_dis: discharging efficiency
14  %  e_chg: charging efficiency
15  %  Q_max: battery capacity
16  %  I_dis: max. discharge current
17  %  I_chg: max. charge current
18
19  %% Calculate
20  %  Battery efficiency
21  eta = inpars.e_dis*(P_in > 0) + inpars.e_chg*(P_in <=0);
22  %  Battery resistance
23  R_BT = (P_in > 0).*interp1(inpars.SOC_list,inpars.R_dis,SOC_in) +
        ...
24          (P_in <= 0).*interp1(inpars.SOC_list,inpars.R_chg,SOC_in);
25  % Current limits
26  I_max = inpars.I_dis*(P_in > 0) + inpars.I_chg*(P_in <= 0);
27  % Battery voltage
28  U_BT = interp1(inpars.SOC_list,inpars.U_ocv,SOC_in);
29  % Calculate Battery current
```

```
30 I_BT = eta.*(U_BT-sqrt(U_BT.^2-4*R_BT.*P_in))./(2*R_BT);
31 % Set to real values
32 I_BT = (I_BT + conj(I_BT))/2;
33
34 %% Overload detection
35 error = 0;
36 %  Infeasibility array
37 Inf_BT = (U_BT.^2 < 4*R_BT.*P_in) + (abs(I_BT) > I_max);
38 %  Set error
39 if(Inf_BT~=0)
40     fprintf('Battery overload error\n')
41     error = 1;
42 end
43
44
45
46 SOC = -I_BT/(inpars.Q_max*3600/ds)+SOC_in;
47
48
49 hdmf_BT_out = struct('I_BT',I_BT,'error_BT',error,'Inf_BT',Inf_BT,'
    SOC',...
50               SOC);
```

Literaturverzeichnis

Back, Michael [2005]. *Prädiktive Antriebsregelung zum energieoptimalen Betrieb von Hybridfahrzeugen*. Band 2 Auflage. Universitätsverlag Karlsruhe. ISBN 3866440316. (Zitiert auf Seite 4.)

Back, Michael, Matthias Simons, Frank Kirschbaum, und Volker Krebs [2002]. *Predictive Control of Drivetrains*. http://www.nt.ntnu.no/users/skoge/prost/proceedings/ifac2002/data/content/02257/abstract.htm. (Zitiert auf Seite 4.)

Bellman, Richard E. [2010]. *Dynamic Programming*. Princeton University Press. ISBN 0691146683. (Zitiert auf Seite 27.)

Foster, Dan [2011]. *GPX 1.1 Schema Documentation*. http://www.topografix.com/gpx/1/1/. (Zitiert auf Seite 19.)

Guzzella, L. und A. Amstutz [2005]. *The QSS Toolbox Manual*. Dissertation, Institut für Mess- und Regeltechnik. http://www.idsc.ethz.ch/Downloads/qss. (Zitiert auf Seiten 3 and 16.)

Guzzella, Lino und Antonio Sciaretta [2005]. *Vehicle Propulsion Systems*. Springer-Verlag Berlin Heidelberg New York. ISBN 3540251952. (Zitiert auf Seite 3.)

Schneider, Gerhard und Hrvatin Mikolcić [1972]. *Einführung in die Methode der dynamischen Programmierung*. R. Oldenbourg Verlag München Wien. ISBN 3486393510. (Zitiert auf Seite 28.)

Stiegler, Markus [2008]. *Entwurf einer vorausschauenden Betriebsstrategie für parallele hybride Antriebsstränge*. Dissertation, Universität Ulm, Germany. http://vts.uni-ulm.de/doc.asp?id=6501. (Zitiert auf Seite 43.)

Sundström, Olle [2009]. *Optimal Control and Design of Hybrid-Electric Vehicles*. Dissertation, Eidgenössische Technische Hochschule, Zürich, Switzerland. http://e-collection.library.ethz.ch/eserv/eth:272/eth-272-02.pdf. (Zitiert auf Seite 43.)

Sundström, Olle und Lino Guzzella [2009]. *A Generic Dynamic Programming MATLAB® Function*. In *Control Applications, (CCA) & Intelligent Control, (ISIC), 2009 IEEE*, Seite 1625. doi:10.1109/CCA.2009.5281131. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5281131. (Zitiert auf Seite 30.)

Wang, Liuping [2009]. *Model Predictive Control System Design and Implementation Using MATLAB®*. Springer-Verlag London Limited. ISBN 1848823304. (Zitiert auf Seite 43.)