Diplomarbeit zum Thema

Auf Dynamischer Programmierung basierende Nichtlineare Modellprädiktive Regelung für LKW

Fakultät für Mathematik und Informatik,
Universität Heidelberg

Alexander Buchner
Januar 2010

betreut durch: Dr. Sebastian Sager

Zusammenfassung

Die *Dynamische Programmierung* ist eine auf dem Gebiet der Optimierung weit bekannte Methode zum Lösen der dort auftretenden Probleme. Wir untersuchen in dieser Arbeit, wie gut sich diese Methode eignet, um auch gemischt-ganzzahlige Optimierungsprobleme aus der Fahrzeugtechnik zu behandeln, in denen zusätzlich dynamische Aspekte per Differentialgleichungssystem modelliert sind.

Konkret nutzen wir die *Dynamische Programmierung*, um einen LKW auf einer vorgegebenen Strecke optimal zu steuern. Dabei ergeben sich vielfältige Möglichkeiten, wenn auf per *GPS* verfügbare Streckendaten zurückgegriffen werden kann. Die Steuerungen umfassen hierbei neben kontinuierlichen Größen auch die diskrete Wahl des optimalen Ganges, was eine besondere Herausforderung an Algorithmen darstellt, die gewöhnlich für komplexe dynamische Systeme eingesetzt werden.

Das verwendete Modell eines LKW ergänzen wir um sinnvolle physikalische Nebenbedingungen und verschiedene frei skalierbare Zielfunktionskomponenten, wie beispielsweise Kraftstoffverbrauch und Anzahl der Schaltvorgänge.

Der von uns mithilfe der *Dynamischen Programmierung* entwickelte Algorithmus nutzt die Struktur dieses Modells in besonderem Maße aus. Um analysieren zu können, welche Gestalt eine global optimale Fahrt annimmt und wie abhängig das Optimum von verschiedenen – teilweise wählbaren – Größen ist, haben wir diesen Algorithmus in Software umgesetzt. Ferner haben wir in diese Software Konzepte der *Nichtlinearen Modellprädiktiven Regelung* einfließen lassen, so dass optimale Steuerungen für einen *Gleitenden Horizont* beliebiger Länge erzeugt werden können.

Eines der Ergebnisse der softwaregestützten Analyse ist die Beobachtung, dass ein überraschend kurzer Prädiktionshorizont ausreicht, um das globale Optimum annähernd zu erreichen. Bei einem Vergleich unserer Ergebnisse mit denen einer direkten Methode ist außerdem zu beobachten, dass die *Dynamische Programmierung* aufgrund der algorithmischen Komplexität lediglich beschränkt einsetzbar ist und im Online-Kontext nur relativ kleine Modelle gehandhabt werden können.

Jedoch stellt sich die *Dynamische Programmierung* insgesamt aufgrund der Erzeugung global optimaler Lösungen als ein sehr mächtiges Werkzeug heraus, das u.a. als Bewertungskriterium für die Ergebnisse anderer Optimalsteuerungsalgorithmen dienen kann.

Danksagung

Zuallererst möchte ich *Professor Dr. Dr. h.c. Hans Georg Bock*, *Dr. Johannes P. Schlöder* und *Dr. Sebastian Sager* dafür danken, dass sie durch eine Vielzahl an Vorlesungen und Seminaren mein Interesse für dieses spannende Gebiet der Mathematik geweckt haben und mich herzlich in ihre Arbeitsgruppe aufgenommen haben.

Ein spezieller Dank geht an *Dr. Sebastian Sager*, der mir während der Zeit, in der diese Arbeit entstand, als Betreuer immer mit Anregungen oder Antworten auf meine Fragen zur Seite stand.

Auch *Christian Kirches* hatte immer ein offenes Ohr für mich und meine Probleme. Ihm gebührt großer Dank.

Für das Korrekturlesen dieser Arbeit möchte ich mich bei *Timur Bircok*, *Michael Engelhart*, *Arlette Haas*, *Florian Kehrle*, *Christian Kirches*, *Dr. Sebastian Sager* und *Robert Schwarz* bedanken.

Ich bedanke mich bei meinen Zimmergenossen *Hendrik Burgdörfer*, *Michael Engelhart* und *Dr. Swen Rupp* für eine angenehme und unterhaltsame Zeit.

Inhaltsverzeichnis

Kapitel 1

Einleitung

In Zeiten steigender Ressourcenknappheit spielt sowohl aus ökologischer als auch ökonomischer Sicht die Minimierung des Treibstoffverbrauchs eines Fahrzeuges eine immer größer werdende Rolle. Dies gilt insbesondere für Lastkraftwagen mit hohem zulässigen Gesamtgewicht. Gerade für Firmen aus dem Bereich der Logistik ist der Treibstoffverbrauch der firmeneigenen Flotte ein Hauptkostenpunkt, den es des zukünftigen wirtschaftlichen Erfolges wegen zu minimieren gilt. Weiterhin gehört der Verkehrssektor zu einem der Hauptverursacher für Treibhausgasemissionen.

Aus diesen Gründen werden große Anstrengungen unternommen, Fahrzeuge herzustellen, die während des Betriebs immer energieeffizienter arbeiten.

Ein weiterer – und gemeinhin unterschätzter – Ansatz zur Senkung des Kraftstoffverbrauchs ist die Anpassung des Fahrverhaltens. Bestimmen oft jahrelang eingeübte Verhaltensmuster die Fahrweise eines Fahrzeugführers, ist es schwer dieses mitunter kostenintensive Verhalten zu ändern. Das in Medien häufig als *vorausschauendes Fahren* propagierte Verhalten in Bezug auf Steigungen und Kurven würde eine erste Verbesserung darstellen. Durch das Verarbeiten externer und exakter Informationen über die Strecke – wie etwa GPS-Daten – könnte eine solche vorausschauende Fahrweise sogar noch optimiert werden. Es ist allerdings nicht davon auszugehen, dass ein menschlicher Fahrer diese Daten während der Fahrt zu einer mathematisch optimalen Steuerung verwerten kann.

An dieser Stelle sind computergestützte Verfahren geeigneter, objektiv sinnvolle Fahrweisen zu bestimmen. Diese Verfahren stützen sich dabei auf ein mathematisches Modell eines Fahrzeuges. Werden zusätzlich Ziele, wie die oben angesprochene Minimierung des Treibstoffverbrauchs oder das Halten einer Wunschgeschwindigkeit, mathematisch formuliert, kann ein daraus entstehendes Optimalsteuerungsproblem durch geeignete Algorithmen gelöst werden, um die Steuerung zu erhalten, die die formulierten Ziele am besten erfüllt.

In diesem Zusammenhang denkbar sind beispielsweise erweiterte Tempomaten, die nicht nur in der Lage sind, eine Geschwindigkeit per variabler Kraftstoffzufuhr zu halten, sondern zusätzlich auch die Gangwahl so zu beeinflussen, dass das Fahrzeug eine vorausliegende Strecke – den jeweiligen Zielen entsprechend – optimal zurücklegen kann.

Die Mathematik der Optimalsteuerungen als Teilgebiet der Numerik ist je nach Be-

schaffenheit des Problems unterschiedlich gut erforscht. So stellt die ganzzahlige, diskrete Wahl eines Ganges in Verbindung mit kontinuierlichen Variablen, wie die der Kraftstoffzufuhr, eine Klasse von Problemen dar, die bis heute Herausforderungen an existierende Algorithmen stellt.

In dieser Arbeit untersuchen wir, inwiefern sich die Methode der *Dynamischen Programmierung* zum Lösen der gerade beschriebenen Optimalsteuerungsprobleme zur Steuerung eines LKW eignet und wie sich deren Ergebnisse im Vergleich zu einem alternativen Algorithmus verhalten.

Weiterhin beschäftigen wir uns mit der Methode der *Nichtlinearen Modellprädiktiven Regelung* und ihrer Wirkung in Verbindung mit der *Dynamischen Programmierung*.

Bei diesen Untersuchungen wird wie folgt vorgegangen:

- Wir beginnen in Kapitel 2 damit, allgemeine Optimalsteuerungsprobleme herzuleiten und zwei Algorithmen zur Lösung dieser Probleme – zum einen die oben bereits genannte *Dynamische Programmierung*, zum anderen die *direkte Mehrzielmethode* – vorzustellen, zu diskutieren und einander gegenüberzustellen.
- Kapitel 3 widmet sich dem Vorhandensein ganzzahliger Variablen, was – wie bereits erwähnt – einen Spezialfall der Optimalsteuerung darstellt. An dieser Stelle gehen wir außerdem darauf ein, wie die im vorherigen Kapitel vorgestellten Algorithmen diesen Fall behandeln bzw. wie sie sich – wenn nötig – erweitern lassen.
- Als letztes rein theoretisches Kapitel folgt Kapitel 4, in welchem die von den zuvor erwähnten Algorithmen unabhängige Technik der *Nichtlinearen Modellprädiktiven Regelung* vorgestellt wird.
- Darauf folgt in Kapitel 5 die Modellierung eines LKW. Das in dieser Arbeit verwendete Modell wird detailliert erläutert. Des Weiteren stellen wir die Bedingungen vor, die für den LKW gelten sollen, und legen die verwendete Zielfunktion dar, um mit Hilfe ihrer mathematischen Formulierungen das Optimalsteuerungsproblem aufzustellen.
- Die algorithmische Umsetzung der auf das LKW-Modell angepassten *Dynamischen Programmierung*, d.h. die im Zuge dieser Arbeit entstandene Software namens *TruckDP*, zeigen wir inklusive Erweiterungen in Kapitel 6 auf.
- In Kapitel 7 stellen wir dann letztendlich die Ergebnisse vor, die wir mittels *TruckDP* berechnet haben, setzen sie in Relation zu Ergebnissen der *direkten Mehrzielmethode* und analysieren die Abhängigkeit der Ergebnisse von Modellgrößen und algorithmischen Einstellungsparametern.
- Wir beschließen die Arbeit mit Kapitel 8, in dem wir alle Ergebnisse zusammenfassen und einen Ausblick auf weitere mögliche Forschungsaspekte in diesem Bereich liefern.

Kapitel 2

Optimalsteuerung

Im Rahmen dieser Arbeit werden Optimalsteuerungsprobleme gelöst, deren mathematische Behandlung in diesem Kapitel untersucht werden soll. Diese Ausführungen bilden dabei die Grundlage für die darauf folgenden Kapitel 3 und 4 dieser Arbeit und sind von konkreten Modellen unabhängig.

2.1 Problemformulierung

Wir stellen zunächst eine allgemeine Formulierung eines nichtlinearen Optimierungsproblems vor:

$$\min_{y \in \mathbb{R}^{n_y}} \quad f(y) \tag{2.1a}$$

$$\text{s.t.} \quad g(y) = 0, \tag{2.1b}$$

$$h(y) \geq 0. \tag{2.1c}$$

Hierbei ist $f : D \to \mathbb{R}$ die *Zielfunktion*, $g : D \to \mathbb{R}^{n_e}$ sind n_e *Gleichungsnebenbedingungen* und $h : D \to \mathbb{R}^{n_i}$ beschreiben n_i *Ungleichungsnebenbedingungen*. $D \subseteq \mathbb{R}^{n_y}$ ist der Definitionsbereich aller dieser Funktionen. Wir nehmen an, dass die Funktionen f, g und h mindestens zweimal stetig differenzierbar sind. Hier sei kurz erwähnt, dass sich Maximierungsprobleme auf die o.g. Form zurückführen lassen, da die Maximierung einer Zielfunktion $f(y)$ der Minimierung von $-f(y)$ entspricht.

Im weiteren Verlauf dieses Abschnitts werden wir diese Formulierung um eine feinere Einteilung der Variablen erweitern.

Wir definieren nun *Zustände*

$$x_i : [t_0, t_f] \to \mathbb{R},\ t \mapsto x_i(t), \quad t_0, t_f \in \mathbb{R}, \quad i \in \{1, \ldots, n_x\}, \tag{2.2}$$

die die Komponenten der Lösung einer gewöhnlichen Differentialgleichung

$$\dot{x}(t) = f(t, x(t)), \quad x = (x_1, \ldots, x_{n_x})^T, \quad t \in [t_0, t_f], \tag{2.3a}$$

$$x(t_0) = x_0 \tag{2.3b}$$

darstellen. Die rechte Seite dieser Differentialgleichung kann außer von der Zeit t und den *differentiellen Zuständen* $x(t)$ auch von frei wählbaren Steuerfunktionen[1]

$$u_i : [t_0, t_f] \to \mathbb{R}, t \mapsto u_i(t), \quad i \in \{1, \dots, n_u\} \tag{2.4}$$

und Parametern $p \in \mathbb{R}^{n_p}$ abhängen. Diese Parameter können im Modell sowohl auf einen konstanten Wert fixiert als auch zur Optimierung freigegeben sein. Im zweiten Fall heißen sie auch *Steuergrößen*. Somit kann man (2.3) zu

$$\dot{x}(t) = f(t, x(t), u(t), p) \tag{2.5}$$

erweitern. Zusätzlich ist es möglich, dass Nebenbedingungen in Form von vorgegebenen Randwerten existieren. Diese schreiben wir als

$$r_e(x(t_0), x(t_f), p) = 0, \tag{2.6a}$$
$$r_i(x(t_0), x(t_f), p) \geq 0. \tag{2.6b}$$

Eventuell vorgegebene Anfangswerte der differentiellen Zustände in der Form $x(t_0) = x_0$ werden in den Bedingungen (2.6a) formuliert.

Betrachten wir nun noch Pfadbeschränkungen der Form

$$g(t, x(t), u(t), p) \geq 0, \tag{2.7}$$

so erhalten wir das folgende allgemeine Optimalsteuerungsproblem:

$$\min_{x,u,p} \quad \Phi(x(t), u(t), p) \tag{2.8a}$$

$$\begin{aligned}
\text{s.t.} \quad \dot{x}(t) &= f(t, x(t), u(t), p), \quad t \in [t_0, t_f] && \text{(2.8b)}\\
0 &= r_e(x(t_0), x(t_f), p), && \text{(2.8c)}\\
0 &\leq r_i(x(t_0), x(t_f), p), && \text{(2.8d)}\\
0 &\leq g(t, x(t), u(t), p), \quad t \in [t_0, t_f]. && \text{(2.8e)}
\end{aligned}$$

Die Zielfunktion $\Phi(\cdot)$ kann hier als *Mayer-Term*

$$\Phi_M = \Phi(t_f, x(t_f), p), \tag{2.9}$$

der nur zum Endzeitpunkt t_f ausgewertet wird, *Lagrange-Term*

$$\Phi_L = \int_{t_0}^{t_f} L(t, x(t), u(t), p) dt \tag{2.10}$$

oder einer Summe aus beidem vorliegen.

[1] Auf eine eventuelle Ganzzahligkeit der Steuerfunktionen gehen wir in Kapitel 3 ein.

Wir weisen daraufhin, dass zur Optimierung freigegebene Parameter p auch als konstante Funktionen aufgefasst werden können, die man in den Steuerungsvektor $u(t)$ aufnehmen kann. Alternativ können auch zusätzliche Zustände $x(t_0) = p;\ \dot{x}(t) = 0$ für die Parameter definiert werden. Aus diesem Grund verzichten wir in ab hier verwendeten Notationen auf die explizite Abhängigkeit von p.

Um diese Art von Optimalsteuerungsproblemen zu lösen, existieren einige Verfahren, von denen wir in den nächsten beiden Abschnitten zwei Vertreter vorstellen wollen. Es werden beide Verfahren vorgestellt, da wir in dieser Arbeit berechnete Optimalsteuerungen verschiedener Algorithmen vergleichen wollen. Zum einen handelt es sich um die auf *Bellman* zurückgehende *Dynamische Programmierung* [3] und zum anderen um die von *Bock und Plitt* in [7] entwickelte *direkte Mehrzielmethode* (engl. *direct multiple shooting method*). Im nächsten Abschnitt stellen wir zunächst die *Direkte Mehrzielmethode* vor und zeigen, wie wir das dabei entstehende Problem auf die in Gleichung (2.1) beschriebene Form bringen können.

2.2 Direkte Mehrzielmethode

Beim *Multiple Shooting* müssen die Funktionen $x(t)$ und $u(t)$, die Elemente eines unendlich-dimensionalen Funktionenraums sind, parametrisiert bzw. diskretisiert werden, um sie numerisch handhaben zu können. Hierzu wird der Zeithorizont $[t_0, t_f]$ in endlich viele Teilintervalle $I_i := [\tau_i, \tau_{i+1}],\ i \in \{0, \ldots, m-1\}$ unterteilt, deren Grenzen wir mit

$$t_0 = \tau_0 < \tau_1 < \ldots < \tau_m = t_f, \quad m \in \mathbb{N} \tag{2.11}$$

bezeichnen. An die Längen $h_i := \tau_{i+1} - \tau_i$ der Teilintervalle werden lediglich die Bedingungen $h_i > 0$ gestellt. Auch ist nicht gefordert, dass die Zeitgitter für die Diskretisierung der Steuerungen und der Parametrisierung der Zustände übereinstimmen müssen, es bringt jedoch Vorteile bezüglich der Separabilität der Zielfunktion, weswegen wir ohne Beschränkung der Allgemeinheit ab jetzt annehmen, dass die Gitter übereinstimmen. Dies kann man z.B. durch Vereinigung der jeweiligen τ_i auf eine gemeinsame Unterteilung erreichen. Das so entstehende gemeinsame Gitter stellt somit eine Verfeinerung der beiden ursprünglichen Gitter dar.

Die Pfadbeschränkungen aus Gleichung (2.7) werden nur punktweise auf dem Gitter der τ_i ausgewertet, so dass wir hier endlich viele Nebenbedingungen

$$g(\tau_i, x(\tau_i; \tau_i, s_i), \phi_i(\tau_i, w_i)) \geq 0, \quad \forall\, i \in \{0, \ldots, m\} \tag{2.12}$$

erhalten. Es ist also möglich, dass die Lösung diese Bedingungen an Stellen, die zwischen den τ_i liegen, nicht erfüllt, sie also verletzt. In vielen Fällen zeigt die Erfahrung, dass diese Verletzung sehr gering ist und akzeptiert werden kann. Für den Fall zu starker Verletzungen existieren allerdings auch Techniken, die es erlauben g kontinuierlich eingehen zu lassen, so dass (2.12) überall erfüllt wird [29].

2.2.1 Diskretisierung der Steuerungen

Auf den soeben definierten Teilintervallen I_i ersetzen wir nun die Steuerungsfunktion u durch *Basisfunktionen*

$$\phi_i : I_i \to \mathbb{R}, t \mapsto \phi_i(t), \quad i \in \{1, \ldots, m\}. \tag{2.13}$$

Diese können z.B. stückweise konstante oder stückweise lineare Funktionen sein, für die in der Regel keine Stetigkeitsbedingungen an den Stellen τ_i gefordert werden. Bei linearen ϕ_i kann allerdings die Forderung der Stetigkeit auftreten. Allgemein müssen sie sich jedoch durch endlich viele Parameter w_i parametrisieren lassen. Somit erhalten wir eine Funktion

$$u(t)\Big|_{I_i} = \phi_i(t, w_i), \quad w_i \in \mathbb{R}^{k_i}, \tag{2.14}$$

die ebenfalls nur von endlich vielen Parametern abhängt, die – zusammengefasst als Vektor w – als Variable in das Optimalsteuerungsproblem eingehen.

2.2.2 Parametrisierung der Zustände

Wie oben bereits erwähnt, benutzen wir für die parametrisierten Zustände das gleiche Gitter, das auch für die Steuerung benutzt wurde. Weiterhin führen wir für den Zustandsvektor x $m+1$ Variablen $s_0, \ldots, s_m,\ s_i \in \mathbb{R}^{n_x}$ ein, die als Anfangswerte des Differentialgleichungssystems an den Stellen $\tau_i,\ i \in \{0, \ldots, m-1\}$ sowie als Endwert für $i = m$ dienen. Die *Anfangswertprobleme* sind somit alle von der Form

$$\dot{x}(t; \tau_i, s_i) = f(t, x_i(t), \phi_i(t, w_i)), \quad t \in [\tau_i, \tau_{i+1}], \tag{2.15a}$$

$$x(\tau_i; \tau_i, s_i) = s_i. \tag{2.15b}$$

Um diese zu lösen, können fehlerkontrollierte Ein- oder Mehrschrittverfahren eingesetzt werden [1].

Die so berechneten Zustände x besitzen – im Gegensatz zur gesuchten Lösung – möglicherweise Unstetigkeitsstellen bei $t = \tau_i$. Um sich nun dem Originalproblem (2.8) wieder zu nähern, führen wir *Schließbedingungen*

$$s_{i+1} = x(\tau_{i+1}; \tau_i, s_i), \quad \forall i \in \{0, \ldots, m-1\} \tag{2.16}$$

ein, die die Stetigkeit der Lösungstrajektorie garantieren.

Die Zielfunktion soll auf jedem Intervall I_i separat ausgewertet werden können und muss sich in

$$\int_{t_0}^{t_f} L(t, x(t), \phi(t, w))dt = \sum_{i=0}^{m-1} L_i(\tau_{i+1}) \tag{2.17}$$

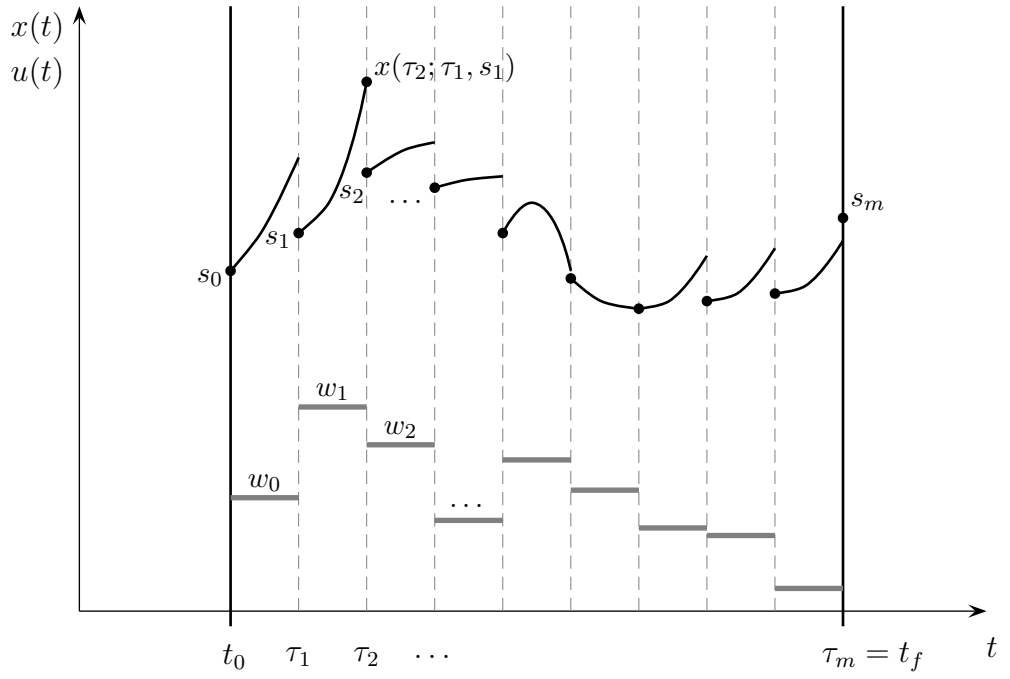

Abbildung 2.1: Mehrzielmethode mit stückweise konstanter Steuerung

mit

$$L_i(\tau_{i+1}) = \int_{\tau_i}^{\tau_{i+1}} L(t, x(t; \tau_i, s_i), \phi_i(t, w_i))dt \tag{2.18}$$

zerlegen lassen.

Wenn wir jetzt noch die Anfangswerte und Randbedingungen auf die neuen Variablen s_i transformieren, erhalten wir für diese weitere Bedingungen:

$$r_e(s_0, s_m) = 0, \tag{2.19a}$$
$$r_i(s_0, s_m) \geq 0, \tag{2.19b}$$
$$s_0 = x_0. \tag{2.19c}$$

Fügen wir die Gleichungen (2.16) bis (2.19) zusammen und sammeln unsere Variablen in einem Vektor

$$y := (s_0, w_0, \ldots, s_{m-1}, w_{m-1}, s_m), \quad s_i \in \mathbb{R}^{n_x}, \quad w_i \in \mathbb{R}^k, \tag{2.20}$$

resultiert daraus folgendes *Nichtlineare Programm* (NLP):

$$\min_{y} \quad \sum_{i=0}^{m-1} L_i(\tau_{i+1}) \tag{2.21a}$$

$$\text{s.t.} \quad 0 = s_{i+1} - x(\tau_{i+1}; \tau_i, s_i) \qquad \forall\, i \in \{0, \dots, m-1\}, \tag{2.21b}$$

$$0 \leq g(\tau_i, x(\tau_i; \tau_i, s_i), \phi_i(\tau_i, w_i)), \qquad \forall\, i \in \{0, \dots, m\}, \tag{2.21c}$$

$$0 = r_e(s_0, s_m), \tag{2.21d}$$

$$0 \leq r_i(s_0, s_m), \tag{2.21e}$$

$$0 = s_0 - x_0. \tag{2.21f}$$

Wir haben das ursprüngliche Problem (2.8) nun also in ein Optimierungsproblem transformiert, dessen Lösung für $m \to \infty$ gegen die Lösung von (2.8) konvergiert und zusätzlich nur endlich viele Optimierungsvariablen und endlich viele Nebenbedingungen besitzt. Es hat somit die Form (2.1) eines *NLPs*. Diese Art von Problemen kann man beispielsweise mit einem *SQP*-Verfahren (engl. *sequential quadratic programming*) lösen.

2.2.3 SQP-Verfahren

In aller Kürze wollen wir beschreiben, wie man das oben hergeleitete *NLP* mit einem *SQP-Verfahren* lösen kann. Grundlegendes zu *SQP-Verfahren* findet man in [18, 30] und speziell bei Verwendung der *Mehrzielmethode* in [26].

Aus dem vorherigen Abschnitt erhalten wir ein *NLP*, welches zusammengefasst als

$$\min_{y} \quad f(y) \tag{2.22a}$$

$$\text{s.t.} \quad g(y) = 0, \tag{2.22b}$$

$$h(y) \geq 0 \tag{2.22c}$$

geschrieben werden kann. f, g und h sind im allgemeinen Fall nichtlineare Funktionen. Wir setzen aber voraus, dass die Funktionen mindestens zweimal stetig differenzierbar sind, da wir für dieses Verfahren die ersten und zweiten Ableitungen benutzen werden. Man versucht sich der Lösung y^* iterativ zu nähern, d.h. man berechnet ausgehend von Anfangswerten y^0 weitere Werte durch Iteration

$$y^{k+1} = y^k + t^k \Delta y^k, \quad t^k \in (0, 1]. \tag{2.23}$$

Das Inkrement Δy^k wird über die Lösung eines *Quadratischen Programms* (QP)

$$\min_{\Delta y} \quad \frac{1}{2} \Delta y^T H^k \Delta y + \nabla f(y^k)^T \Delta y \tag{2.24a}$$

$$\text{s.t.} \quad g(y^k) + \nabla g(y^k)^T \Delta y = 0, \tag{2.24b}$$

$$h(y^k) + \nabla h(y^k)^T \Delta y \geq 0 \tag{2.24c}$$

Aktion	Laufzeitkomplexität
Berechnung der Hessematrix	$O(m^2(n_x + n_q)^3)$
Berechnung der Nebenbedingungen	$O(m^2(n_x + n_q)^3)$
QP-Löser, Initialisierung	$O((mn_u + n_x)^3)$
QP-Löser, pro Iteration	$O((mn_u + n_x)^2)$
Zurückgewinnung der eigentlichen Lösung	$O(mn_x{}^2)$

Tabelle 2.1: Laufzeitkomplexität für *Condensing* und einen QP-Löser für dicht besetzte Probleme in Abhängigkeit von der Anzahl der Mehrziel-Intervalle m, der Steuerungen n_u und der Zustände n_x.

bestimmt. $\nabla f(y^k)$ bezeichnet den *Gradienten* der Funktion f an der Stelle y^k. Man lässt dieses Verfahren in der Regel enden, wenn sich y^{k+1} nur noch geringfügig von seinem Vorgänger y^k unterscheidet. H^k ist hier die *Hessematrix* der *Lagrange-Funktion* im Schritt k. An dieser Stelle existiert auch die Möglichkeit eine Näherung der Hessematrix zu benutzen oder diese über mehrere Schritte beizubehalten, anstatt sie in jeder Iteration neu zu berechnen. Gründe für die Verwendung einer Näherung sind u.a. die verkürzte Laufzeit zur Erzeugung der Näherung im Gegensatz zur Berechnung der exakten Matrix aber auch die Möglichkeit positiv definite Matrizen zu verwenden, falls die exakte Hessematrix diese Eigenschaften nicht erfüllt. Der zuletzt genannte Grund führt zu besseren theoretischen Eigenschaften des Verfahrens. Die *Schrittweite* t^k kann z.B. über eine *Liniensuche* bestimmt werden.

Es sei noch kurz erwähnt, dass das *QP*, das man aus der *direkten Mehrzielmethode* erhält, eine spezielle Struktur besitzt. Die Matrix der Nebenbedingungen wird aufgrund der Einführung der s_i als zusätzliche Variablen groß, jedoch aber *dünn besetzt*, d.h. sie enthält viele Null-Einträge. Durch *Condensing* [7, 27] kann die Matrix in eine kleinere, dicht besetzte Matrix transformiert werden, die dem QP-Löser dann als Eingabe dient.

Falls das Problem mehr Steuerungen als Zustände besitzt, ist der Gewinn, den das *Condensing* bringt, aber eher gering.

2.2.4 Komplexität

Die Laufzeit eines *SQP-Verfahrens* setzt sich zum einen aus der Anzahl der Iterationen, die benötigt werden, und zum anderen aus der Laufzeit des *QP-Lösers* zusammen.

Die Anzahl der Iterationen, bis die optimale Lösung gefunden ist, die beim iterativen Lösen eines *Quadratischen Programms* auftritt, hängt maßgeblich von der Beschaffenheit des Problems selbst ab. Falls z.B. die linearisierten Nebenbedingungen linear abhängig sind, kann es vorkommen, dass ein periodisches Durchlaufen (engl. *cycling*) der Iterierten auftritt. In diesem Fall sind die QPs ohne geeignete Strategie zur Vermeidung von *cycling* nicht in endlicher Zeit lösbar, da immer wieder die gleiche Sequenz der Iterationen

durchlaufen wird, ohne sich der Lösung des QPs zu nähern. Auch wenn die Nebenbedingungen nicht linear abhängig sind, der Winkel zwischen ihnen aber sehr spitz ist, wirkt sich das negativ auf die Anzahl der Iterationen aus.

Die Laufzeiten im allgemeinen Fall sind in Tabelle 2.1 übersichtlich dargestellt.

Das *SQP-Verfahren* an sich konvergiert *lokal quadratisch*, also lediglich in einer geeigneten Umgebung der tatsächlichen Lösung. Daher sind gute Startwerte sowohl für die Qualität der Lösung als auch für die Laufzeit, um die Lösung zu erhalten, wichtig. Die oben angesprochene *Liniensuche* ist eine Strategie, die darauf abzielt, das Konvergenzverhalten zu globalisieren.

Für weiterführende Informationen sei an dieser Stelle auf die Literatur ([12, 13, 15, 16, 38]) verwiesen.

2.3 Dynamische Programmierung

2.3.1 Optimalitätsprinzip

Die *Dynamische Programmierung* (DP) basiert auf dem in [3] formulierten *Optimalitätsprinzip* von *Bellmann*, welches im Original folgendermaßen lautet:

> „An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."

Wenn wir dies nun auf das von uns in Gleichung (2.8) formulierte Optimalsteuerungsproblem übertragen, die optimale Zustandstrajektorie mit $x^*(t)$ und die dazugehörige optimale Steuerung mit $u^*(t)$, jeweils $t \in [t_0, t_f]$, bezeichnen, so bedeutet dieses Optimalitätsprinzip, dass für jedes $t_1 \in [t_0, t_f]$ die Teilsteuerung $u^*(t), t \in [t_1, t_f], 0 \leq t_1 \leq t_f$ optimal ist, um den Zustand $x^*(t_1)$ in $x^*(t_f)$ zu überführen.

Dass diese Aussage richtig ist, lässt sich leicht per Widerspruch verifizieren. Wäre die Teilsteuerung $u^*(t), t \in [t_1, t_f]$ nicht optimal um $x^*(t_1)$ in $x^*(t_f)$ zu überführen, gäbe es eine andere Steuerung $v(t), t \in [t_1, t_f]$ mit einem kleineren Zielfunktionswert auf diesem Abschnitt. Daraus würde aber folgen, dass die Steuerung

$$v^*(t) := \begin{cases} u^*(t), & \text{falls } t \in [t_0, t_1] \\ v(t), & \text{falls } t \in [t_1, t_f] \end{cases} \tag{2.25}$$

insgesamt einen kleineren Zielfunktionswert als $u^*(t)$ liefern würde. Dies wäre aber ein Widerspruch zur angenommenen Optimalität von $u^*(t)$.

Dass eine optimale Trajektorie aus optimalen Teiltrajektorien besteht, ist also offensichtlich. Man beachte aber, dass die Umkehrung der Aussage nicht gilt. Wenn Steuerungen $u_i,\ i \in \{0, \ldots, N-1\}$ gegeben sind, die jeweils auf dem Horizont $[t_i, t_{i+1}]$ optimal

sind, so ist die Steuerung

$$u'(t) := \begin{cases} u_0(t), & \text{falls } t \in [t_0, t_1] \\ u_1(t), & \text{falls } t \in [t_1, t_2] \\ \vdots & \\ u_{N-1}(t), & \text{falls } t \in [t_{N-1}, t_N] \end{cases} \tag{2.26}$$

nicht zwangsläufig optimal für den gesamten Horizont $[t_0, t_f]$.

2.3.2 Herleitung

Die *Dynamische Programmierung* war ursprünglich für die Lösung zeitdiskreter Optimalsteuerungsprobleme vorgesehen und basiert daher auf einem Modell, welches in zeitlich diskreten Abschnitten abläuft. Es wurde also keine kontinuierliche Steuerungsfunktion u gesucht, sondern zu mehreren festen Zeitpunkten eine Entscheidung unter endlich vielen Alternativen. Daher teilen wir den Horizont $[t_0, t_f]$ in N Teilstücke auf. Das dynamische System wird durch *Übergangsfunktionen*

$$x_{k+1} = f_k(x_k, u_k), \quad k \in \{0, 1, \ldots, N-1\} \tag{2.27}$$

beschrieben. Wie im vorherigen Abschnitt bezeichnet $x_k \in S_k \subseteq \mathbb{R}^{n_x}$ einen Zustand und $u_k \in C_k \subseteq \mathbb{R}^{n_u}$ eine Steuerung. Die Steuerung u_k nimmt, abhängig vom Zustand x_k, Werte aus $U_k(x_k) \subseteq C_k$ an.

Die Zielfunktion, die unserem Optimierungsproblem zu Grunde liegt, ist bezüglich der Zeit additiv. Der Beitrag zur Zielfunktion,[2] der im Abschnitt k entsteht, wird mit $g_k(x_k, u_k)$ bezeichnet. Zusätzlich kann es Endkosten $g_N(x_N)$ geben. Daraus ergeben sich die Gesamtkosten, die der Zielfunktion

$$J(\cdot) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k) \tag{2.28}$$

entsprechen. Wir definieren Abbildungen

$$\mu_i : S_k \to C_k, x_k \mapsto u_k, \quad i \in \{0, \ldots, N-1\}, \tag{2.29}$$

die Zustände auf Steuerungen abbilden und fassen diese in einem Steuerungsvektor

$$\pi = \{\mu_0, \ldots, \mu_{N-1}\} \tag{2.30}$$

zusammen. Es interessieren uns diejenigen π, die für das Problem zulässig sind. Steuerungsvektoren nennen wir *zulässig*, wenn sie Zustände x_k per u_k in gültige Zustände x_{k+1} überführen:

$$f_k(x_k, \mu_k(x_k)) = x_{k+1} \in S_{k+1} \quad \forall k \in \{0, \ldots, N-1\}. \tag{2.31}$$

[2] Im folgenden werden wir auch von den Kosten bzw. der Kostenfunktion reden.

Die Kosten, die für festes x_0 und festes π entstehen, erhält man durch Auswertung der Zielfunktion:

$$J_\pi(x_0) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k)). \tag{2.32}$$

Für festes x_0 ist die optimale Steuerung π^* somit diejenige, die diese Kosten minimiert,

$$J_{\pi^*}(x_0) = \min_{\pi \in \Pi} J_\pi(x_0), \tag{2.33}$$

wobei Π die Menge aller zulässigen Steuerungsvektoren ist. $J_{\pi^*}(x_0)$ sind die *optimalen Kosten*, die für den Startwert x_0 entstehen. Für freies x_0 ist daher das Minimum

$$\min_{x_0 \in S_0} \left(\min_{\pi \in \Pi} J_\pi(x_0) \right) \tag{2.34}$$

zu finden.

Nun stellt sich natürlich die Frage, wie diese Minima berechnet werden können. Eine *Enumeration* ist bei sehr kleinen Problemen natürlich möglich, dennoch ist zu beachten, dass die Anzahl der Möglichkeiten, den Vektor π zu bilden, sehr schnell wächst. Schon bei nur einer Steuerung ist

$$|\Pi| \approx \prod_{i=0}^{N-1} |U_i(x_i)| \tag{2.35}$$

nur in gewissen Grenzen überschaubar.

Hier kommen wir auf das in Abschnitt 2.3.1 erwähnte *Optimalitätsprinzip* zurück. Dort haben wir gesehen, dass die „Endstücke“ der optimalen Steuerung auf jeden Fall auch optimal sein müssen. Dabei ist es egal, wo wir den Anfang dieses Stückes definieren. Die Idee, die das Optimalitätsprinzip vorgibt, ist nun also zuerst Lösungen eines Optimalsteuerungsproblems auf einem kleinen Abschnitt – dem Ende – zu finden und dieses dann „nach vorne“ auf größere Horizonte zu erweitern, bis man t_0 erreicht.

2.3.3 Algorithmus

Wir stellen jetzt zunächst den Algorithmus der *Dynamischen Programmierung* vor und beweisen anschließend seine Korrektheit.

Ab hier verwenden wir die Notation $\hat{x}_j \in S_k$ für den j-ten Diskretisierungspunkt des Zustands x zur Zeit k, wobei x_j weiterhin die Lösung von 2.27 bezeichnet.

In $J_i(\hat{x}_j)$ werden wir die Fortsetzungskosten (engl. *cost-to-go*) abspeichern, die auf dem Abschnitt $[t_i, t_f]$ entstehen, wenn man von $\hat{x}_j$ aus startet. Diese Kosten initialisieren wir im Endzeitpunkt $t_f = t_N$ mit den Endkosten

$$J_N(\hat{x}_j) = g_N(\hat{x}_j) \quad \forall \hat{x}_j \in S_N. \tag{2.36}$$

Die Hauptidee des Algorithmus ist folgende Rekursion: Man berechnet, mit $k = N-1$ beginnend,

$$J_k(\hat{x}_j) = \min_{u_i \in U_k(\hat{x}_j)} \left(g_k(\hat{x}_j, u_i) + J_{k+1}(f_k(\hat{x}_j, u_i))\right) \quad \forall \hat{x}_j \in S_k, \tag{2.37}$$

um die Fortsetzungskosten zu tabellieren. Hierbei erfolgt die Minimierung über alle u_i, die von $\hat{x}_j$ aus zulässig sind. Anschließend berechnet man J_k für $k = N-2, \ldots, 0$.

Satz 1 (Bertsekas [4])**.** *Für jeden Anfangswert $\hat{x}_j \in S_0$ stimmt das hier berechnete $J_0(\hat{x}_j)$ mit den optimalen Kosten $J_{\pi^*}(\hat{x}_j)$ aus Gleichung (2.33) überein.*

Beweis. Für alle Steuerungsvektoren $\pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$ und alle $k \in \{0, 1, \ldots, N-1\}$ sei $\pi^k = \{\mu_k, \mu_{k+1}, \ldots, \mu_{N-1}\}$. Bezeichne x_k einen beliebigen zulässigen Zustand $\hat{x}_j \in S_k$ und alle x_l mit $l > k$ die aus (2.27) erhaltenen Zustände. Seien $J_k^*(x_k)$ die optimalen Kosten für das $(N-k)$-stufige Problem mit Anfangswert x_k auf $[t_k, t_N]$, also[3]

$$J_k^*(x_k) = \min_{\pi^k} \left(g_N(x_N) + \sum_{i=k}^{N-1} g_i(x_i, \mu_i(x_i))\right). \tag{2.38}$$

Für $k = N$ definieren wir $J_N^*(\hat{x}_j) = g_N(\hat{x}_j) \quad \forall \hat{x}_j \in S_N$. Wir werden nun per Induktion nach Abschnittsindex k zeigen, dass die Funktionen J_k^* mit den Funktionen J_k aus dem DP-Algorithmus übereinstimmen.

Als Induktionsanfang dient $J_N^* = J_N = g_N$. Dies gilt bereits per Definition.

Nehmen wir also an, es gäbe ein k, für das $J_{k+1}^*(x_j) = J_{k+1}(x_j)$ für alle $x_j \in S_{k+1}$ gelte, so gilt mit $\pi^k = (\mu_k, \pi^{k+1})$ für alle $x_k \in S_k$

$$\begin{aligned}
J_k^*(x_k) &= \min_{(\mu_k, \pi^{k+1})} \left(g_k\Big(x_k, \mu_k(x_k)\Big) + g_N(x_N) + \sum_{i=k+1}^{N-1} g_i\Big(x_i, \mu_i(x_i)\Big)\right) && \text{(2.39a)}\\
&= \min_{\mu_k} \left(g_k\Big(x_k, \mu_k(x_k)\Big) + \min_{\pi^{k+1}} \left(g_N(x_N) + \sum_{i=k+1}^{N-1} g_i\Big(x_i, \mu_i(x_i)\Big)\right)\right) && \text{(2.39b)}\\
&= \min_{\mu_k} \left(g_k\Big(x_k, \mu_k(x_k)\Big) + J_{k+1}^*\Big(f_k(x_k, \mu_k(x_k))\Big)\right) && \text{(2.39c)}\\
&= \min_{\mu_k} \left(g_k\Big(x_k, \mu_k(x_k)\Big) + J_{k+1}\Big(f_k(x_k, \mu_k(x_k))\Big)\right) && \text{(2.39d)}\\
&= \min_{u_k \in U_k(x_k)} \left(g_k\Big(x_k, u_k\Big) + J_{k+1}\Big(f_k(x_k, u_k)\Big)\right) && \text{(2.39e)}\\
&= J_k(x_k). && \text{(2.39f)}
\end{aligned}$$

[3] Mit $x_{k+1} = f_k(x_k, \mu_k(x_k)) \quad \forall k \in \{0, \ldots, N-k-1\}$.

Im ersten Schritt ziehen wir die Minimierung über π^{k+1} in die Klammer, da der erste Term nur von μ_k abhängt. Im zweiten Schritt wird die Definition von J^*_{k+1} und im dritten die Induktionshypothese benutzt. □

Somit ist gezeigt, dass die Funktionen $J_k(\hat{x}_j)$ die optimalen Fortsetzungskosten vom Zustand $\hat{x}_j \in S_k$ aus beschreiben. Führt man die Rekursion bis $k = 0$ durch, so erhält man, bei freiem Anfangswert, für jeden Anfangswert x_0 die optimalen Kosten. Für festes x_0 müssen die Berechnungen der $J_0(\hat{x}_j)$ für $\hat{x}_j \neq x_0$ somit nicht durchgeführt werden.

Um den optimalen Steuerungsvektor π zu erhalten, speichert man sich beim Durchlauf des DP-Algorithmus bei der Berechnung der J_k die Steuerung ab, die diese jeweils minimiert. So kann man von seinem Anfangswert aus, durch Auslesen dieser Steuerung und anschließende Auswertung der Übergangsfunktion $f_k(x_k, u_k)$ zum nächsten Zustand auf der optimalen Trajektorie gelangen, bei der man diese Prozedur wiederholt.

Das kontinuierliche Analogon zum diskreten DP-Algorithmus ist die *Hamilton-Jacobi-Bellman-Gleichung*. Ausgehend von einem durch eine Differentialgleichung

$$\dot{x}(t) = f(x(t), u(t)), \quad t_0 \leq t \leq t_f \tag{2.40}$$

und eine angepasste Kostenfunktion

$$h(x^u(t_f)) + \int_{t_0}^{t_f} g(x^u(t), u(t))dt \tag{2.41}$$

gegebenen System ist die *Hamilton-Jacobi-Bellman-Gleichung* eine partielle Differtialgleichung, deren Lösung ebenfalls eine Funktion $J^*(t, x)$ ist, die optimale Fortsetzungskosten im kontinuierlichen Fall beschreibt. Für weiterführende Informationen siehe z.B. [4].

2.3.4 Diskretisierung

Falls man sich, wie in dieser Arbeit geschehen, dafür entscheidet die *Dynamische Programmierung* auf ein kontinuierliches Problem anzuwenden, muss man zwangsläufig die kontinuierlich vorliegenden Größen diskretisieren. Dies betrifft zum einen sicherlich die unabhängige Variable[4] aber auch Zustände und Steuerungen können nun nicht mehr beliebige Werte aus $\mathbb{R}^n$ annehmen.

Falls z.B. ein Zustand im ursprünglichen Modell reellwertig war, so muss man sich nun für ein Diskretisierungsgitter entscheiden, da zum „Zeitpunkt“ k alle $J_k(x_k)$ berechnet werden müssen. Hierbei muss man darauf achten, dass durch die Diskretisierung auch das Ergebnis

$$x_{k+1} = f_k(x_k, u_k) \tag{2.42}$$

[4]Bisher war dies immer die Zeit t.

nicht mehr notwendigerweise auf dem gewählten Diskretisierungsgitter liegt. Da dieser Zustand bzw. dessen Fortsetzungskosten J_{k+1} aber benötigt werden, muss man eine Möglichkeit finden, diese zu approximieren. Dies kann z.B. dadurch erreicht werden, dass man das Ergebnis der Übergangsfunktion auf den nächsten Gitterpunkt „rundet“ und dann diese Fortsetzungskosten verwendet. Alternativ ist natürlich auch eine Interpolation der Fortsetzungskosten der benachbarten Gitterpunkte durchführbar. Im mehrdimensionalen Fall ist dies allerdings mit Aufwand verbunden, der mit jeder weiteren Dimension exponentiell wächst.

Wie wir im nächsten Abschnitt sehen werden, spielt die Feinheit der jeweiligen Diskretisierungen natürlich auch für die Betrachtung der Komplexität eine Rolle. Generell ist bei der Wahl der Gitter darauf zu achten, dass man einen Kompromiss zwischen einer möglichst guten Approximation der reellwertigen Variablen und einer nicht allzu hohen Laufzeit des Algorithmus findet. Arbeitet man mit nichtäquidistanten Gittern der Zustände, ist darauf zu achten, dass das Gitter eher in dem Bereich fein ist, in dem die Lösung zu erwarten ist. Falls keinerlei Kenntnisse über eine eventuelle Lösung vorliegen, so kann das Problem erst mal auf einem groben Gitter gelöst werden um dann im Lösungsbereich zu verfeinern.

2.3.5 Komplexität

Betrachten wir nun die Komplexität der *Dynamischen Programmierung*. Wir bezeichnen mit $\alpha_i(k),\ i \in \{1,\dots,n\}$ die Anzahl der Gitterpunkte der n Zustände zum Zeitpunkt k und mit $\beta_j(k),\ j \in \{1,\dots,m\}$ die Anzahl der m diskreten Steuerungen, ebenfalls zum Zeitpunkt k. Das Zustandsgitter beinhaltet dann insgesamt

$$\sum_{k=0}^{N} \prod_{i=1}^{n} \alpha_i(k) \tag{2.43}$$

Gitterpunkte über den ganzen Zeithorizont $[t_0, t_f]$. An jedem dieser Gitterpunkte müssen für die Minimierung der Kostenfunktion $J_k(x_k)$

$$\prod_{j=1}^{m} \beta_j(k) \tag{2.44}$$

Steuerungen betrachtet werden. Schätzen wir die einzelnen $\alpha_i(k)$ und $\beta_j(k)$ mit

$$\alpha := \max_i \alpha_i(k), \quad \beta := \max_j \beta_j(k) \tag{2.45}$$

nach oben ab, so ergibt sich für die Rechenzeit τ eine Proportionalität zu

$$\tau \sim N \cdot \alpha^n \cdot \beta^m. \tag{2.46}$$

Hier sieht man, dass die Laufzeit exponentiell mit den Dimensionen der Zustände und der Steuerungen wächst. *Bellman* spricht in diesem Zusammenhang von „*curse of dimensionality*“ (dt. etwa „*Fluch der Dimensionalität*“). Günstigere Konditionen ergeben sich dagegen in Bezug auf die Horizontlänge N, da diese lediglich linear in die Komplexität eingeht. Ebenso geht die Feinheit der Diskretisierungen der Zustände und Steuerungen linear ein. Dies ist also nicht der Hauptfaktor, der lange Laufzeiten verursacht, aber ein doppelt so feines Gitter für einen einzigen Zustand oder eine einzige Steuerung ruft auch eine Verdopplung der Rechenzeit hervor.

2.4 Vergleich der beiden Methoden

Durch die grundlegend verschiedenen Herangehensweisen an die Probleme ergeben sich auch einige unterschiedliche Eigenschaften der beiden vorgestellten Methoden. Hauptunterschiede sind sicherlich bei den Eigenschaften der Lösungen zu finden, die diese Algorithmen produzieren.

Die *direkte Mehrzielmethode* ist ein Verfahren, das ausgehend von Startwerten x^0 für die differentiellen Zustände auf dem zeitlichen Diskretisierungsgitter anhand von Ableitungen versucht, Werte x^{k+1} durch Iteration zu erzeugen, die das Zielfunktional verbessern. Da die Iterationen in Richtung des lokal steilsten Abstiegs der Zielfunktion erfolgen, führt dieses Vorgehen zu einem *lokalen Optimum*. Falls das Problem nicht *konvex* ist, ist es also durchaus möglich, dass es andere (lokale) Optima gibt, die einen besseren Zielfunktionswert liefern.

Im Gegensatz dazu deckt die *Dynamische Programmierung* durch die Tabellierung aller Fortsetzungskosten den ganzen Suchraum ab und man erhält ein *globales Optimum*.

Durch diese Tabellierung hat die *Dynamische Programmierung* einen enormen Speicherbedarf, der im gleichen Maße wie die Laufzeit wächst. Aufgrund der Proportionalität zur Laufzeit ist auch der Speicherbedarf schon bei Problemen mit wenigen Zuständen eventuell ein Problem.[5]

Ein interessanter Unterschied ergibt sich auch, wenn man die Bedeutung der Beschränkungen betrachtet, die in den Optimierungsproblemen formuliert sind. Wird die *direkte Mehrzielmethode* verwendet, erschweren viele Nebenbedingungen die Suche nach zulässigen Lösungen und – wie in Abschnitt 2.2.4 erläutert – unter bestimmten Bedingungen können sie die Laufzeit negativ beeinflussen oder im schlechtesten Fall sogar die Lösung eines Unterproblems verhindern.

Bei der *Dynamischen Programmierung* hingegen können die Nebenbedingungen erheblich dazu beitragen, dass die Laufzeit des Algorithmus abnimmt. Das liegt daran, dass der Algorithmus für jeden Zustandsknoten jede mögliche Steuerung absucht um das Optimum zu finden. Durch Beschränkungen der Zustände und Steuerungen verkleinert sich

[5] In Kapitel 6 werden Techniken vorgestellt, mit denen es zumindest temporär gelingt, den Speicherbedarf zu verkleinern.

der Suchraum mitunter erheblich, was dann zu den kürzeren Laufzeiten führt.

Eine der wenigen Gemeinsamkeiten, den beide Typen von Algorithmen besitzen, ist, dass sie sich beide gut parallelisieren lassen. Bei der *Mehrzielmethode* kann man z.B. die Lösung der Anfangswertprobleme auf allen Intervallen parallel rechnen. Für die *Dynamische Programmierung* ist eine Parallelisierung auch sehr gut zu vollführen, da man die Suche nach der optimalen Fortsetzung für alle Zustandsknoten eines Zeitgitterpunkts gleichzeitig durchführen kann.

Der Vorteil, der aus der Berücksichtigung jedes Zustandsknotens erwächst, ist, dass man zusätzlich auch einen optimalen *Regelsatz* abseits der optimalen Lösungstrajektorie erhält. Falls man – z.B. durch Störungen im System – zu einem Zeitpunkt t_k von der optimalen Trajektorie abkommt, muss kein neues Optimierungsproblem gelöst werden, da die optimale Fortsetzung für alle Zustände zu allen Zeiten bereits bekannt ist und nachgeschlagen werden kann. Im Fall der *Mehrzielmethode* ist hier auf dem kürzeren Horizont $[t_k, T_N]$ erneut zu optimieren.

Den Preis, den man dafür zu zahlen hat, ist die Komplexität der *Dynamischen Programmierung*. Wie wir gesehen haben, wächst sowohl die Laufzeit als auch der Speicherplatz exponentiell mit den Dimensionen des Problems. Dadurch lassen sich praktisch nur sehr kleine Modelle mit vertretbarer Diskretisierung lösen.

Aufgrund der Tatsache, dass die Mehrzielmethode mit *Condensing* für Probleme mit wenigen Zuständen und vielen Steuerungen nicht so effektiv ist wie im umgekehrten Fall, fällt der Vergleich der Laufzeiten in diesem Fall noch am positivsten für die *Dynamische Programmierung* aus. Wie wir später sehen werden, fällt das in dieser Arbeit behandelte Modell eines LKW in genau diese Kategorie.

Kapitel 3

Ganzzahlige optimale Steuerung

3.1 Motivation

Im letzten Kapitel haben wir die Theorie der *Optimalsteuerung* hergeleitet, die nötig ist, um die Probleme, die in dieser Arbeit auftreten, lösen zu können. Dabei haben wir allerdings eine Besonderheit der hier auftretenden Probleme zunächst ausgelassen: die Ganzzahligkeitsbedingungen, die an manche Steuerungen gestellt werden. Typischerweise treten diese Art von Variablen z.B. als Modellierung von Ventilen oder – wie in dieser Arbeit – als Variable für den gewählten Gang auf. Diese Probleme nennen wir *gemischt-ganzzahlige Optimalsteuerungsprobleme* (engl. *mixed-integer optimal control problem*, *MIOCP*).

Die Existenz ganzzahliger Variablen verlangt bei Verwendung der *Dynamischen Programmierung* keine besondere Aufmerksamkeit, da sie auf natürliche Weise mit diskreten Steuerungen umzugehen vermag, was gegenüber der *direkten Mehrzielmethode* einen deutlichen Vorteil darstellt. Denn Letztere lässt den Einsatz geeigneter Verfahren nötig werden, um die Ganzzahligkeit der Lösung zu garantieren.

Verwendet man die *direkte Mehrzielmethode*, um das im vorherigen Kapitel vorgestellte Optimalsteuerungsproblem (2.8) zu lösen, so ist nicht auf Anhieb erkennbar, wie man aus der Lösung dieses Problems – ohne die Forderung der Ganzzahligkeit – eine Lösung generiert, die die genannte zusätzliche Bedingung erfüllt *und* auch für dieses veränderte Problem optimal ist oder zumindest in der Nähe der Lösung liegt.

Bei der Lösung der abgeschwächten Probleme ohne Ganzzahligkeitsbedingungen können verschiedene Szenarien auftreten. Bei vielen Problemen, in denen die Minimierung der Zeit gefordert wird – z.B. zeitoptimales Fahren mit einem Fahrzeug – hat die Lösung eine *Bang-Bang*-Struktur. Dies bedeutet, dass die Steuerung im Lösungspunkt bis auf Diskretisierungsfehler ausschließlich Werte an ihrer unteren bzw. oberen Grenze annimmt. Im Fall der zeitoptimalen Fahrt entspräche das einer Vollbremsung vor einer Kurve, direkt gefolgt von einer Vollgasphase.

Allerdings gibt es auch Fälle, in denen die optimale Steuerung Werte zwischen diesen Grenzen annimmt. Es handelt sich hierbei um einen *singulären Bogen*. In diesen Fällen gestaltet sich die Suche nach einer ganzzahligen Steuerung, die die Zielfunktion minimiert, mitunter wesentlich schwieriger. Ein einfaches Beispiel ist hier das *Lotka-*

Volterra-Modell zur Beschreibung zweier Spezies, von denen die eine Räuber und die andere Beute modelliert.[6]

Zur Verdeutlichung stellen wir uns ein Problem vor, in dessen zugrunde liegendem Modell ein Ventil Beachtung findet, das lediglich geschlossen ($u = 0$) oder geöffnet ($u = 1$) sein kann. Daraus ergäbe sich die Bedingung, dass

$$u(t) \in \{0, 1\} \quad \forall t \tag{3.1}$$

sein muss. Würde man diese Forderung auf

$$u(t) \in [0, 1] \tag{3.2}$$

abschwächen[7], kann man versuchen dieses Problem mit den in Kapitel 2 vorgestellten Methoden zu lösen. Angenommen, man erhalte als Optimalsteuerung den Vektor der diskretisierten Steuerungen

$$u = (u_1, \ldots, u_m), \quad 0.5 \leq u_i \leq 0.6 \quad \forall i \in \{1, \ldots, m\}. \tag{3.3}$$

So wäre *in diesem Fall* eine einfache Rundung auf die nächste ganze Zahl für jedes Intervall vermutlich keine gute *Heuristik*. Hier würde man also auf die Steuerung

$$u_i \equiv 1 \quad \forall i \tag{3.4}$$

runden, was wahrscheinlich nicht in der Nähe der Lösung des *MIOCP* liegt. Vor allem aber kann es sein, dass in diesem Fall eventuelle Pfadbeschränkungen verletzt würden.

Falls für das relaxierte Problem dagegen eine Steuerung berechnet wird, die sich schon in der Nähe einer ganzen Zahl befindet, so besteht für diese einfache Heuristik sicherlich eine größere Wahrscheinlichkeit, eine gute Lösung zu produzieren.

Im Folgenden werden wir eine Methode beschreiben, die im allgemeinen Fall eingesetzt werden kann.

3.2 Problemformulierung

Um im Folgenden zwischen den kontinuierlichen und den diskretwertigen Steuerungen unterscheiden zu können, werden wir – wie bisher – die kontinuierlichen Steuerungen mit $u(t) \in \mathbb{R}^{n_u}$ bezeichnen, während wir für die diskreten Steuerungen $v(t) \in \mathbb{R}^{n_v}$ schreiben werden.

Das Differentialgleichungssystem, das die Dynamik des zugrunde liegenden Modells beschreibt, schreibt sich somit in diesem Kapitel als

$$\dot{x}(t) = f(x(t), u(t), v(t)), \quad t \in [t_0, t_f] \tag{3.5}$$

[6] Genaueres über dieses Modell lässt sich u.a. in [31] und [34] nachlesen.
[7] Wir nennen dies im Folgenden *Relaxierung*.

mit

$$v(t) \in \Omega := \{v^1, v^2, \ldots, v^{n_\omega}\}, \quad t \in [t_0, t_f], \;\; n_\omega \in \mathbb{N}. \tag{3.6}$$

$\Omega \subset \mathbb{R}^{n_v}$ ist somit endlich.

Unter einer – oben schon kurz angesprochenen – *Relaxierung* verstehen wir eine Abschwächung einer gegebenen Bedingung

$$x \in X, \quad X = \{x^1, \ldots, x^n\}, \quad n \in \mathbb{N} \tag{3.7}$$

zu

$$x \in \operatorname{conv} X. \tag{3.8}$$

$\operatorname{conv} X$ ist dabei die *konvexe Hülle* einer Menge X und definiert als

$$\operatorname{conv} X = \left\{ \sum_{i=1}^{n} \alpha_i \cdot x_i \,\middle|\, x_i \in X, n \in \mathbb{N}, \sum_{i=1}^{n} \alpha_i = 1, \alpha_i \geq 0 \right\}. \tag{3.9}$$

Die Relaxierung der Bedingung (3.6) ist somit

$$v(t) \in \operatorname{conv} \Omega, \quad t \in [t_0, t_f], \tag{3.10}$$

wobei $\operatorname{conv} \Omega \subset \mathbb{R}^{n_v}$ die *konvexe Hülle* der Punkte v^i ist. Im eindimensionalen Fall $n_v = 1$ gilt also:

$$\operatorname{conv} \Omega = [a, b] \subset \mathbb{R}, \quad a := \min_i v^i, \quad b := \max_i v^i. \tag{3.11}$$

Die Klasse der Probleme, die wir in diesem Kapitel lösen werden, ist Folgende:

$$\min_{x,u,v} \quad \Phi(x(t), u(t), v(t)) \tag{3.12a}$$

$$\begin{aligned}
\text{s.t.} \quad \dot{x}(t) &= f(t, x(t), u(t), v(t)), \quad t \in [t_0, t_f] && \text{(3.12b)}\\
x(t_0) &= x_0, && \text{(3.12c)}\\
0 &= r_e(x(t_0), x(t_f)), && \text{(3.12d)}\\
0 &\leq r_i(x(t_0), x(t_f)), && \text{(3.12e)}\\
0 &\leq g(t, x(t), u(t), v(t)), && \text{(3.12f)}\\
v(t) &\in \Omega. && \text{(3.12g)}
\end{aligned}$$

3.3 Konvexifizierung

Der Begriff der *Konvexifizierung*, den wir in dieser Arbeit gebrauchen werden, bezieht sich immer auf eine *Konvexifizierung* im Raum der binären Steuerungen. Es ist möglich, dass das Problem in allen anderen Variablen weiterhin *nichtkonvex* ist.

Ab hier lassen wir die Pfadbeschränkungen g und die Randbedingungen r_e und r_i aus Gründen der Übersichtlichkeit aus und transformieren zusätzlich die ganzzahlige Steuerung $v(t)$ zu einer binären Steuerfunktion $\omega(t)$. So erhalten wir das binäre, nichtlineare Problem (BN) (nach engl. *binary nonlinear*):

$$\min_{x,u,\omega} \quad \Phi(x(t), u(t), \omega(t)) \tag{3.13a}$$

$$\begin{aligned} \text{s.t.} \quad \dot{x}(t) &= f(t, x(t), u(t), \omega(t)), \quad t \in [t_0, t_f] && (3.13b)\\ x(t_0) &= x_0, && (3.13c)\\ \omega(t) &\in \{0,1\}^{n_\omega}. && (3.13d) \end{aligned}$$

Ersetzen wir nun (3.13d) durch $\omega(t) \in [0,1]^{n_\omega}$, schwächen also die Bedingung dahingehend ab, dass $\omega(t)$ für jede vektorielle Komponente zwischen 0 und 1 liegen darf, erhalten wir ein neues Problem, welches wir mit (RN) (nach engl. *relaxed nonlinear*) bezeichnen:

$$\min_{x,u,\omega} \quad \Phi(x(t), u(t), \omega(t)) \tag{3.14a}$$

$$\begin{aligned} \text{s.t.} \quad \dot{x}(t) &= f(t, x(t), u(t), \omega(t)), \quad t \in [t_0, t_f] && (3.14b)\\ x(t_0) &= x_0, && (3.14c)\\ \omega(t) &\in [0,1]^{n_\omega}. && (3.14d) \end{aligned}$$

Dies wird auch die *innere Konvexifizierung* genannt, da $\omega(t)$ als Funktionsargument (im Inneren) von $f(\cdot)$ relaxiert wird. Alternativ dazu konvexifizieren wir nun durch Linearisierung in ω, indem wir (3.13b) durch

$$\dot{x}(t) = \sum_{i=1}^{n_{\tilde{\omega}}} f(t, x(t), u(t), \omega^j)\ \tilde{\omega}_i(t), \quad \tilde{\omega}(t) \in \{0,1\}^{n_{\tilde{\omega}}} \tag{3.15}$$

ersetzen. Die ω^j sind fest und durchlaufen alle binären Steuerungen. Daher gilt $n_{\tilde{\omega}} = 2^{n_\omega}$. Die relaxierte Funktion $\tilde{\omega}_i(t)$ steht hier außerhalb der Funktion $f(\cdot)$, weshalb man hier auch von der *äußeren Konvexifizierung* spricht.[8]

Um ein Problem zu erhalten, das äquivalent zu (3.13) ist, fügen wir eine *SOS1*-Bedingung hinzu:

$$\sum_{i=1}^{n_{\tilde{\omega}}} \tilde{\omega}_i(t) = 1, \quad \forall t \in [t_0, t_f]. \tag{3.16}$$

[8] Durch die äußere Konvexifizierung wird also ein Problem erzeugt, das nun wesentlich mehr Steuerungen aufweist, was das in Abschnitt 2.2.3 angesprochene *Condensing* ineffektiver macht.

Hieraus entsteht ein in $\tilde{\omega}$ konvexes Problem (BC) (nach engl. *binary convex*):

$$\min_{x,u,\tilde{\omega}} \quad \Phi(x(t), u(t), \tilde{\omega}(t)) \tag{3.17a}$$

$$\text{s.t.} \quad \dot{x}(t) = \sum_{i=1}^{n_{\tilde{\omega}}} f(t, x(t), u(t), \omega^j)\ \tilde{\omega}_i(t), \quad t \in [t_0, t_f] \tag{3.17b}$$

$$x(t_0) = x_0, \tag{3.17c}$$

$$\tilde{w}(t) \in \{0,1\}^{n_{\tilde{\omega}}}, \tag{3.17d}$$

$$\sum_{i=1}^{n_{\tilde{\omega}}} \tilde{\omega}_i(t) = 1. \tag{3.17e}$$

Zwischen den Lösungen von (3.13) und (3.17) existiert eine Bijektion (siehe Theorem 1, unten). Auch zu Problem (3.17) geben wir ein relaxiertes Problem (RC) (nach engl. *relaxed convex*) an. Wir ersetzen (3.17d) durch $\tilde{\omega}(t) \in [0,1]^{n_{\tilde{\omega}}}$ und erhalten

$$\min_{x,u,\tilde{\omega}} \quad \Phi(x(t), u(t), \tilde{\omega}(t)) \tag{3.18a}$$

$$\text{s.t.} \quad \dot{x}(t) = \sum_{i=1}^{n_{\tilde{\omega}}} f(t, x(t), u(t), \omega^j)\ \tilde{w}_i(t), \quad t \in [t_0, t_f] \tag{3.18b}$$

$$x(t_0) = x_0, \tag{3.18c}$$

$$\tilde{\omega}(t) \in [0,1]^{n_{\tilde{\omega}}}, \tag{3.18d}$$

$$\sum_{i=1}^{n_{\tilde{\omega}}} \tilde{\omega}_i(t) = 1. \tag{3.18e}$$

Diese vier Probleme dienen nun als Grundlage des weiteren Abschnitts. Wir stellen zunächst zwei Theoreme vor, die Aussagen über die Vergleichbarkeit von Lösungen der jeweiligen Probleme machen.

Theorem 1 (Vergleich binärer Lösungen)**.** *Falls das Problem (BC) eine Optimallösung $(x^*, u^*, \tilde{\omega}^*)$ mit Zielfunktionswert Φ^{BC} besitzt, existiert eine n_ω-dimensionale Steuerung $\omega^*(\cdot)$, so dass die Trajektorie (x^*, u^*, ω^*) eine Optimallösung von (BN) mit Zielfunktionswert Φ^{BN} ist. Weiterhin gilt*

$$\Phi^{BN} = \Phi^{BC}. \tag{3.19}$$

Die Umkehrung gilt ebenfalls.

Theorem 2 (Vergleich der Lösungen konvexifizierter Probleme)**.** *Sei $(x^*, u^*, \tilde{\omega}^*)$ eine zulässige Lösung für (RC) mit Zielfunktionswert Φ^{RC}. Weiterhin sei $f(x, u^*, \omega)$ für alle zulässigen binären Steuerungen $\omega(\cdot)$ global Lipschitz-stetig bzgl. $x(\cdot)$ (u^* fest). Dann*

gilt: Zu jedem $\epsilon > 0$ existiert eine zulässige Steuerung $\overline{\omega}$ und eine Funktion $\overline{x}$, so dass die dazugehörige Trajektorie $(\overline{x}, u^*, \overline{\omega})$ eine zulässige Lösung des Problems (BC) mit Zielfunktionswert Φ^{BC} ist.
Es gilt:

$$\Phi^{BC} \leq \Phi^{RC} + \epsilon. \tag{3.20}$$

Dieses Theorem sagt also aus, dass man den Zielfunktionswert einer vorgegebenen kontinuierlichen Steuerung beliebig gut mit einer binären Steuerung approximieren kann. Es sei angemerkt, dass dies insbesondere für den Fall gilt, dass $(x^*, u^*, \tilde{\omega}^*)$ die Optimallösung von (RC) ist. Es kann allerdings vorkommen, dass die binäre Steuerung auf einem Zeitintervall sehr oft hin und her schalten muss, um das zu erreichen.

Weiterhin hat dieses Theorem zur Folge, dass man nun Aussagen über die Qualität von berechneten binären Lösungen machen kann, da man durch die kontinuierliche Optimallösung – im Falle einer Minimierung – die maximale untere Schranke erhält. Man kann so beurteilen, wie gut die aktuelle binäre Lösung tatsächlich ist.[9]

3.4 MS MINTOC

In diesem Abschnitt stellen wir zunächst Heuristiken vor, mit deren Hilfe wir aus Lösungen der relaxierten Probleme möglichst gute Lösungen für das binäre Problem erzeugen wollen. Dabei kommen wir auch auf das zu Anfang des Kapitels bereits kurz erwähnte Runden zurück. Danach beschreiben wir das Schema eines Algorithmus namens *MS MINTOC*, der von *Sager et al.* in [36] vorgestellt wurde.

Ab hier werden wir die relaxierten Steuerungen mit $q_{j,i} \in [0,1]$ und die binären mit $p_{j,i} \in \{0,1\}$ bezeichnen. $j \in \{1, \ldots, n_\omega\}$ bezeichnet die vektorielle Komponente ($q = (q_1, \ldots, q_{n_\omega})^T$), während $i \in \{0, \ldots, N-1\}$ den Index des Zeitintervalls I_i angibt.

3.4.1 Adaptive Gitterverfeinerung

Die *Gitterverfeinerung* ist eine Methode, die angewendet werden kann, falls es bei der Lösung des Problems (3.18) auf einem festen, groben Gitter $\mathcal{G}^k$ Intervalle I_k gibt, auf denen die optimale Steuerung keinen ganzzahligen Wert annimmt. Dabei muss die Steuerung gleichzeitig auf dem einen der beiden benachbarten Intervalle I_{k-1} und I_{k+1} Werte an ihrer unteren und auf dem anderen Intervall Werte an ihrer oberen Grenze annehmen. Hier vermutet man also eine *Bang-Bang*-Struktur, deren Übergang durch die Diskretisierung verloren gegangen ist, da das gewählte Gitter zu grob war. Man versucht durch Einfügen zusätzlicher Gitterpunkte τ^* auf dem Intervall I_k diesen vermuteten Schaltpunkt in das aktuelle Diskretisierungsgitter aufzunehmen, um danach das Problem erneut zu lösen. Nun wird erwartet, dass an diesem weiteren Schaltpunkt ein direkter Übergang der

[9] Die Beweise sind in [36] nachzulesen.

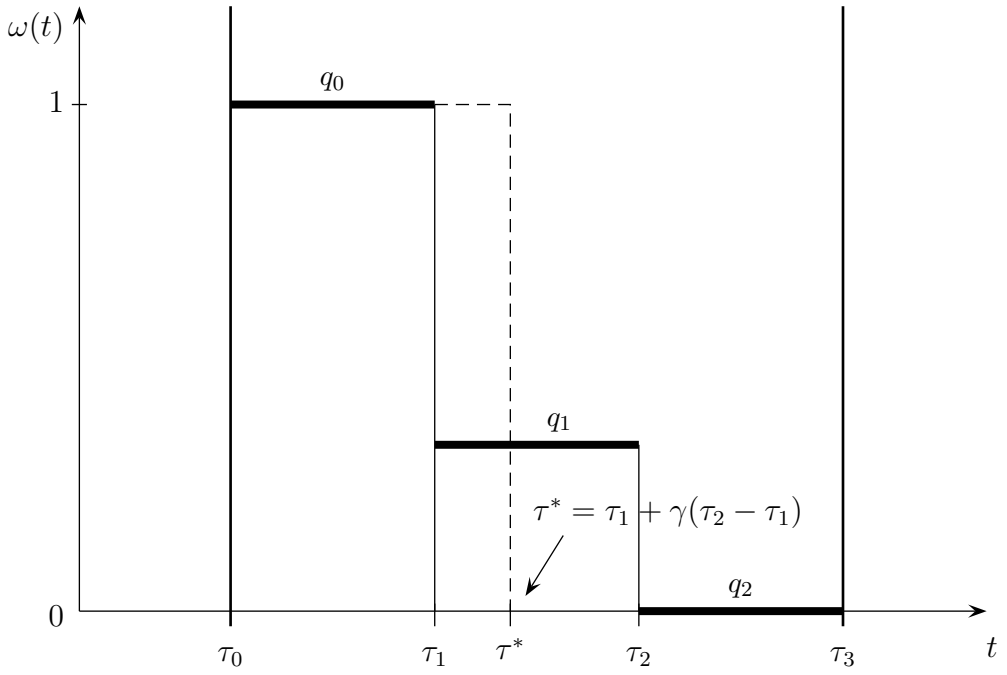

Abbildung 3.1: Gitterverfeinerung

Steuerung stattfindet. Falls dies nicht eintritt, kann diese Prozedur erneut durchgeführt werden.

Abhängig vom Wert q_i kann man folgende Heuristik anwenden[10]:

$$q_i = 0 \Longrightarrow \text{Annahme } \omega_i = 0 \tag{3.21a}$$

$$q_i = 1 \Longrightarrow \text{Annahme } \omega_i = 1 \tag{3.21b}$$

$$q_i \notin \{0, 1\} \Longrightarrow \text{füge zusätzlichen Gitterpunkt } \tau^* \text{ hinzu.} \tag{3.21c}$$

Bereits ganzzahlige Abschnitte werden nicht mehr verfeinert, während wir in (3.21c) einen zusätzlichen Punkt fordern. Wir wollen diesen abhängig von q_i als $\tau^* = \tau_i + \gamma(\tau_{i+1} - \tau_i)$ bestimmen. Abhängig von der Richtung des Übergangs bestimmen wir

$$\gamma = \begin{cases} q_i & \text{falls } q_{i-1} = 1 \text{ und } q_{i+1} = 0 \\ 1 - q_i & \text{falls } q_{i-1} = 0 \text{ und } q_{i+1} = 1 \end{cases} \tag{3.22}$$

und nehmen den neuen Punkt in das aktuelle Gitter auf, d.h. $\mathcal{G}^{k+1} := \mathcal{G}^k \cup \tau^*$. Zur Anschauung dient Abbildung 3.1.

[10] Zur Vereinfachung schildern wir das Vorgehen nur für den eindimensionalen Fall.

Falls auf mehreren benachbarten Intervallen die Steuerung nicht ganzzahlig ist, ist das Wissen über eine optimale binäre Steuerung auf diesen Abschnitten im Normalfall zu gering, um etwas sinnvolleres als eine einfache *Bisektion* mit $\gamma = 0.5$ durchzuführen.

Eine andere Möglichkeit, mit nichtganzzahligen q_i umzugehen, stellt das Runden dar.

3.4.2 Rundungsstrategie

Löst man das relaxierte Problem (3.18)[11] auf einem festen Gitter, erhält man eine relaxierte Optimallösung. Im besten Fall weist diese bereits eine *Bang-Bang*-Struktur auf. Diese ganzzahlige Lösung muss optimal für Problem (3.17) sein, da sie optimal für das relaxierte Problem (3.18) ist, in dem die Menge der variablen Steuerungen eine Obermenge der Menge der ganzzahligen Steuerungen ist. In der Regel wird die durch Relaxierung gewonnene Lösung aber nicht ganzzahlig sein. Hier besteht die Möglichkeit, die relaxierte Optimallösung durch Runden der einzelnen $q_{j,i}$ auf jedem Intervall I_i zur Ganzzahligkeit zu „zwingen“.

Neben dem simplen Runden

$$p_{j,i} = \begin{cases} 1 & \text{falls } q_{j,i} \geq 0.5 \\ 0 & \text{sonst} \end{cases} \tag{3.23}$$

jedes einzelnen Wertes unabhängig von den anderen, gibt es eine alternative Strategie, die *Sager et al. „sum up rounding“* nennen [36]. Hierbei betrachtet man beim Runden nicht nur den aktuell zu rundenden Wert, sondern auch die Rundungsentscheidungen, die bisher getroffen wurden:

$$p_{j,i} = \begin{cases} 1 & \text{falls } \sum_{k=0}^{i} q_{j,k} - \sum_{k=0}^{i-1} p_{j,k} \geq 0.5 \\ 0 & \text{sonst} \end{cases} \tag{3.24}$$

Mit den Bezeichnungen

$$\alpha_j(t) := q_{j,i}, \quad \beta_j(t) := p_{j,i}, \quad t \in [t_i, t_{i+1}] \tag{3.25}$$

und

$$\Delta t := \max_{i=0\ldots N-1} \Delta t_i = \max_{i=0\ldots N-1} \{t_{i+1} - t_i\} \tag{3.26}$$

ist das Ergebnis dieser Strategie folgendes

Theorem 3. *Seien Funktionen $\alpha : [t_0, t_f] \to [0,1]^{n_\omega}$ und $\beta : [t_0, t_f] \to \{0,1\}^{n_\omega}$ durch (3.24) und (3.25) gegeben. Weiterhin sei Δt definiert wie in (3.26). Dann gilt*

$$\left\| \int_{t_0}^{t} \beta(\tau) - \alpha(\tau) d\tau \right\| \leq 0.5 \, \Delta t. \tag{3.27}$$

[11] Dafür eignet sich beispielsweise die in Kapitel 2 vorgestellte *direkte Mehrzielmethode*.

Der Beweis ist in [33] zu finden.

Falls – wie im Falle der Konvexifizierung – die *SOS1*-Bedingung aus (3.16) gefordert wird, muss beachtet werden, dass $q_{j,k}$ nicht unabhängig von $q_{i,k}$ gerundet werden kann. Mit Hilfsvariablen

$$\hat{q}_{j,k} := \sum_{k=0}^{i} q_{j,k} - \sum_{k=0}^{i-1} p_{j,k} \tag{3.28}$$

rundet man dann

$$p_{j,i} = \begin{cases} 1 & \text{falls } \hat{q}_{j,i} \geq \hat{q}_{k,i} \; \forall k \neq j \quad \text{und} \quad j < k \; \forall \, k \; : \; \hat{q}_{j,i} = \hat{q}_{k,i} \\ 0 & \text{sonst} \end{cases} . \tag{3.29}$$

Das Ergebnis dieser Strategien erfüllt grundsätzlich die Ganzzahligkeitsbedingungen, nicht unbedingt jedoch vorhandene Pfadbeschränkungen. Falls diese doch erfüllt werden, hat man auf diese Weise überdies eine obere Schranke[12] für das zu lösende Minimierungsproblem erhalten, die sich im *Branch and Bound*-Kontext nutzen lässt. Da diese Strategien lediglich Heuristiken sind, ist das Ergebnis im Regelfall auch nicht optimal, kann aber z.B. als Ausgangssituation der Schaltpunktoptimierung dienen, die im nächsten Abschnitt vorgestellt wird.

3.4.3 Schaltpunktoptimierung

Die *Schaltpunktoptimierung* (engl. *switching time optimization*) ist eine Technik zur Verbesserung einer gegebenen *Bang-Bang*-Lösung, die auf einem festen Gitter berechnet wurde. Diese ist entweder bei der Lösung des relaxierten Problems entstanden oder ist z.B. das Ergebnis einer Rundungsstrategie.

Die Schaltstruktur der Lösung wird hierbei nicht mehr verändert, lediglich die Zeitpunkte τ_i, in denen die Steuerung von 0 auf 1 und umgekehrt „schaltet“, werden jetzt zur Optimierung freigegeben. Man hat also – im eindimensionalen Fall – eine vorgegebene binäre Steuerung

$$p_i = \begin{cases} 1 & \text{falls } i \text{ gerade} \\ 0 & \text{falls } i \text{ ungerade} \end{cases} \tag{3.30}$$

oder umgekehrt. In der Praxis deklariert man die Abschnittslängen $h_k = t_{k+1} - t_k$ als Optimierungsvariablen, siehe [14]. Da sie praktisch zeitunabhängig sind, kann man sie als freie Parameter zu Problem (3.17) hinzufügen. Zusätzlich muss allerdings

$$\sum_{k=0}^{N-1} h_k = t_f - t_0 \tag{3.31}$$

gefordert werden. Zur Verdeutlichung der Vorgehensweise, siehe Abbildung 3.2. Dieser

[12] Wird ein Maximierungsproblem behandelt, erhält man eine untere Schranke.

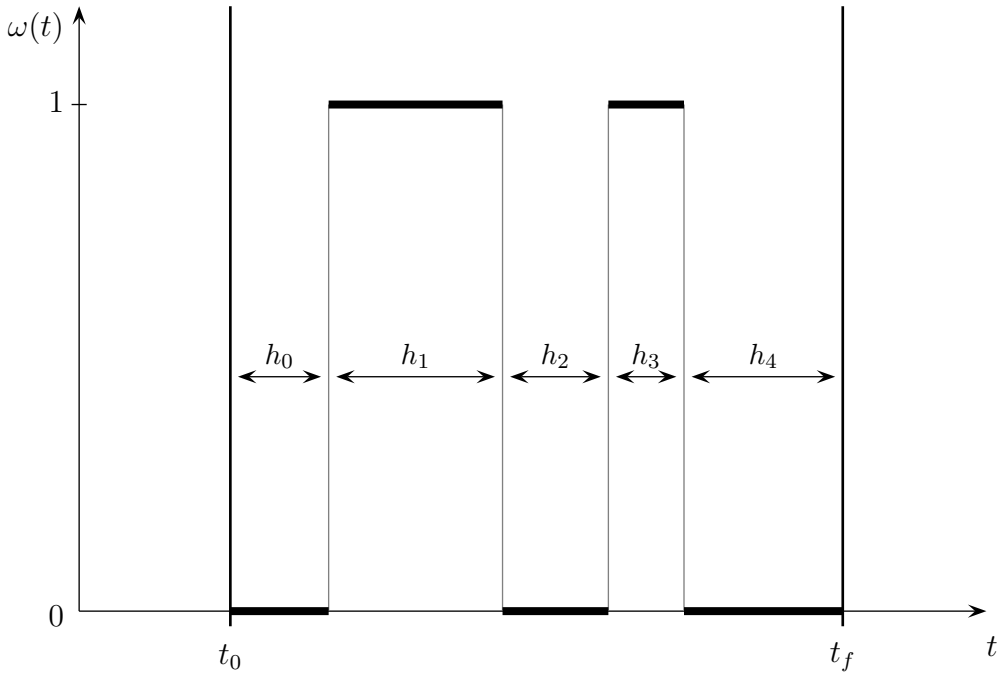

Abbildung 3.2: Schaltpunktoptimierung

Ansatz kann auch auf den Fall mehrdimensionaler Steuerungen erweitert werden. Dies werden wir hier nicht vorstellen, sondern verweisen auf [32] und [36].

Es besteht grundsätzlich die Möglichkeit, die Schaltpunktoptimierung auch als eigenständige Optimierungstechnik einzusetzen. Allerdings entstehen durch diese Technik viele lokale Minima, weshalb eine gute Initialisierung ausschlaggebend für den Erfolg dieser Methode ist. Die in Abschnitt 3.4.2 erläuterte Rundungsstrategie liefert in der Regel gute Startwerte, so dass eine Verbindung dieser beiden Techniken günstig sein kann. Weiterhin muss bei Verwendung der Schaltpunktoptimierung die Schaltstruktur bereits bekannt sein.

In [14] ist ein Anwendungsbeispiel im Automobilbereich zu finden.

3.4.4 Algorithmus

Nachdem wir einige Techniken vorgestellt haben, fügen wir diese nun zusammen und stellen den Algorithmus *MS MINTOC* (*multiple shooting based mixed-integer optimal control*) vor. Die optimale Trajektorie von (3.17) wird hier mit $\mathcal{T}^k = (x^k(\cdot), u^k(\cdot), \tilde{\omega}^k(\cdot))$ bezeichnet.

Algorithmus 1. (MS MINTOC)

1. $k = 0$. *Eingabe: anfängliches Diskretisierungsgitter der Steuerungen* $\mathcal{G}^0$, *Toleranz* $TOL \in \mathbb{R}^+$.

2. *Falls nötig, wird das Problem (3.13) konvexifiziert (Abschnitt 3.3). Man erhält ein Problem der Form (3.17). Dieses wird zu* $\tilde{\omega}(\cdot) \in [0,1]^{n_{\tilde{\omega}}}$ *relaxiert (3.18).*

3. *Wiederhole:*
   a) *Löse relaxiertes Problem auf* $\mathcal{G}^k$. *Erhalte* $\mathcal{T}^k = (x^k(\cdot), u^k(\cdot), \tilde{\omega}^k(\cdot))$ *und den gitterabhängigen optimalen Zielfunktionswert* $\Phi^{REL}_{\mathcal{G}^k}$.
   b) *Falls* $\mathcal{T}^k$ *auf* $\mathcal{G}^k$ $\tilde{\omega}^k(\cdot) \in \{0,1\}^{n_{\tilde{\omega}}}$ *erfüllt, dann STOP.*
   c) *Wende „sum up rounding“ auf* $\tilde{\omega}^k(\cdot)$ *an. Halte* $u^k(\cdot)$ *fest. Erhalte obere Schranke* $\Phi^{BIN}_{\mathcal{G}^k}$ *durch Simulation.*
   d) *Optional: Schaltpunktoptimierung (Abschnitt 3.4.3).*
   e) *Falls* $\Phi^{BIN}_{\mathcal{G}^k} < \Phi^{REL}_{\mathcal{G}^k} + TOL$, *dann STOP.*
   f) *Verfeinere das Gitter* $\mathcal{G}^k$ *(adaptiv).*
   g) $k = k + 1$.

4. *Benutze Bijektion, um Lösung für Problem (3.13) mit Zielfunktionswert* $\Phi^* = \Phi^{BIN}_{\mathcal{G}^k}$ *zu erhalten.*

Um in Schritt 3f) eine adaptive Gitterverfeinerung durchzuführen, speichert man in Schritt 3c) die $q_i \notin \{0,1\}$ ab, um wie in Abschnitt 3.4.1 vorzugehen.

Erfolgreiche Anwendung für Fahrzeugmodelle fand dieser Algorithmus und insbesondere die beschriebene Konvexifizierung in [25] und [35].

Algorithmische Erweiterungen dieser Methoden zur Ausnutzung spezieller Strukturen bei Verwendung der direkten Mehrzielmethode sind in [23] und [24] zu finden.

Kapitel 4

Nichtlineare Modellprädiktive Regelung

Der Grundgedanke aller prädiktiven Regelungen zur Beeinflussung dynamischer Systeme besteht darin, dass man ein gegebenes Optimalsteuerungsproblem nicht einmalig auf einem festen Zeithorizont $[t_0, t_f]$ löst, um dann die berechnete optimale Steuerung auf das System aufzutragen, sondern man versucht das System zu steuern, während es sich in der Ausführung befindet. Anwendung findet diese Vorgehensweise hauptsächlich bei Prozessen, die ständig und ununterbrochen laufen und somit nicht auf einen endlichen Zeithorizont ausgelegt sind. Dazu zählen beispielsweise Heizungs- und Klimaanlagen, die eine konstante Raumtemperatur halten sollen, oder Herstellungsprozesse in der chemischen Verfahrenstechnik. Durch das Berechnen der optimalen Steuerung parallel zum Prozess ist es jederzeit möglich, auf eventuell auftretende Störungen zu reagieren und „gegenzulenken".

Während man bei Heizungs- und Klimaanlagen o.ä. oftmals einfache regelbasierte Steuerungen – z.B. Heizleistung erhöhen, falls die gewünschte Temperatur aktuell unterschritten ist – vorfindet, wird bei komplexeren dynamischen Systemen ein Modell des jeweiligen Systems in die Steuerung miteinbezogen. Bei chemischen Destillationskolonnen zum Beispiel benutzt man ein Differentialgleichungsmodell der ablaufenden Reaktion, um vorausschauend steuern zu können. Angewendet auf das Beispiel der Heizungsanlage könnte man eine Wettervorhersage für die nahe Zukunft in das Modell integrieren und die Steuerung der Heizleistung von den zukünftigen Außentemperaturen abhängig machen. Die zu steuernde zukünftige Raumtemperatur würde man unter Zuhilfenahme des Modells prädizieren, um so eine bzgl. eines Zielfunktionals möglichst gute Steuerung zu berechnen. Diese Berechnung fände hier also – im Gegensatz zu Reglern, die nur auf dem aktuellen Systemzustand basierend eine Entscheidung treffen – *modellprädiktiv* statt.

Im nicht-prädiktiven Fall wird die Steuerungsantwort direkt aus einer Funktionsvorschrift in Abhängigkeit vom aktuellen Zustand gewonnen. Hier kann – wenn gewünscht – eine Tabellierung der Antworten erfolgen. Im Unterschied dazu wird bei einem modellprädiktiven Regler ein Optimalsteuerungsproblem gelöst, um die Steuerungsantwort zu erhalten. Falls alle auftretenden Optimalsteuerungsprobleme lösbar sind, kann theoretisch auch in diesem Fall eine Tabelle der vorher berechneten Antworten aufgestellt werden. In der Praxis scheitert dieses Vorhaben aber häufig daran, dass die Zustandsräume zu groß sind und eine vorherige Tabellierung der Antworten nicht durchführbar

ist.

Die zu beeinflussende Größe – im vorherigen Beispiel war dies die Raumtemperatur – nennt man die *Regelgröße* y. Diese wird direkt durch die *Stellgröße* u beeinflusst, die das Ergebnis der Regelung ist. Eingangsgröße für die Regelung ist zum einen die *Sollgröße* w und zum anderen die zurückgeführte *Regelgröße* y.

Ist das zugrunde liegende Differentialgleichungsmodell linear, also

$$\dot{x}(t) = Ax + Bu \tag{4.1}$$

mit $x \in \mathbb{R}^{n_x},\ u \in \mathbb{R}^{n_u}$ und Matrizen $A \in \mathbb{R}^{n_x \times n_x},\ B \in \mathbb{R}^{n_x \times n_u}$, spricht man von *Linearer Modellprädiktiver Regelung* (engl. *Model Predictive Control*, MPC). Sie wird heutzutage in vielen Bereichen angewendet und auch die dazugehörige Theorie ist sehr gut erforscht.

Im Gegensatz dazu handelt es sich um eine *Nichtlineare Modellprädiktive Regelung* (engl. *Nonlinear Model Predictive Control*, NMPC), wenn sich das System nicht linear modellieren lässt. Ein großes Problem in diesem noch jungen Teilgebiet, an dem aktuell intensiv geforscht wird, stellte lange Zeit dar, dass die Algorithmen für die Lösung des Optimalsteuerungsproblems parallel zum laufenden Prozess nicht schnell genug waren. Infolgedessen wurden viele Systeme durch lineare Differentialgleichungen und insbesondere lineare Nebenbedingungen approximiert und mit Hilfe von linearen Reglern gesteuert. Ist die „Nichtlinearität“ des Systems nicht zu sehr ausgeprägt, zeigt dieses Vorgehen gute Ergebnisse und hat sich über viele Jahre bewährt.

Mittlerweile wurden aber auch für die *Nichtlineare Modellprädiktive Regelung* sehr schnelle Algorithmen entwickelt, die für die Echtzeitanwendung geeignet sind, siehe z.B. [9].

4.1 Gleitender Horizont

Das Prinzip der *Nichtlinearen Modellprädiktiven Regelung* wird in der Regel auf einem *gleitenden Horizont* (engl. *moving horizon*) umgesetzt. Die Länge dieses Horizonts bezeichnen wir mit T. Ausgehend vom zur Zeit t_i gemessenen Systemzustand x_i und dem durch das Modell gegebenen Differentialgleichungssystem

$$\dot{x}(t) = f(t, x(t), u(t), p) \tag{4.2}$$

löst man ein Optimalsteuerungsproblem für den *Prädiktionshorizont* $[t_i, t_i + T]$, das sich wie in (2.8) schreiben lässt.[13] Der erste Teil der optimalen Steuerung auf dem Abschnitt $[t_i, t_i + \delta],\ \delta < T$ wird an das System zurückgegeben, um es zu steuern. Spätestens zur Zeit $t_i + \delta$ wird nun wiederum der Systemzustand erfasst und ein neues Optimierungsproblem auf dem Horizont $[t_i + \delta, t_i + \delta + T]$ gelöst. Der erste Abschnitt $[t_i + \delta, t_i + 2\delta]$ der auf $[t_i + \delta, t_i + \delta + T]$ optimalen Steuerung wird auf das System gegeben. Dieser Zyklus kann

[13] Welcher Algorithmus zur Lösung benutzt wird, ist an dieser Stelle nicht wichtig.

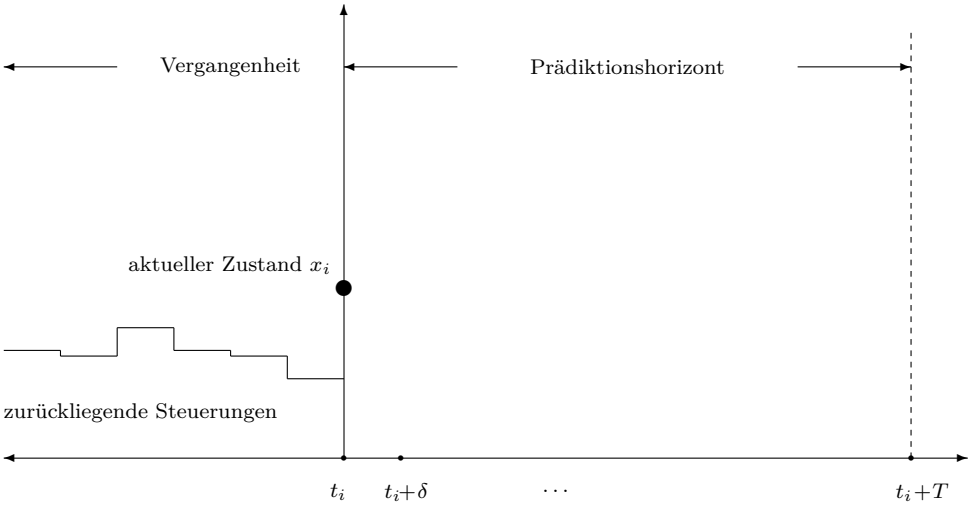

Abbildung 4.1: Ausgangssituation zur Zeit t_i

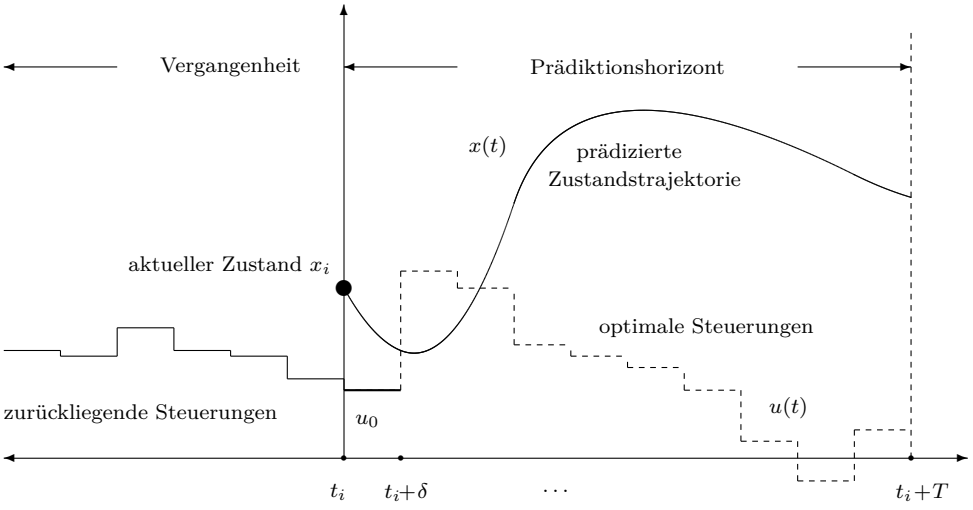

Abbildung 4.2: Berechnung der optimalen Steuerungen

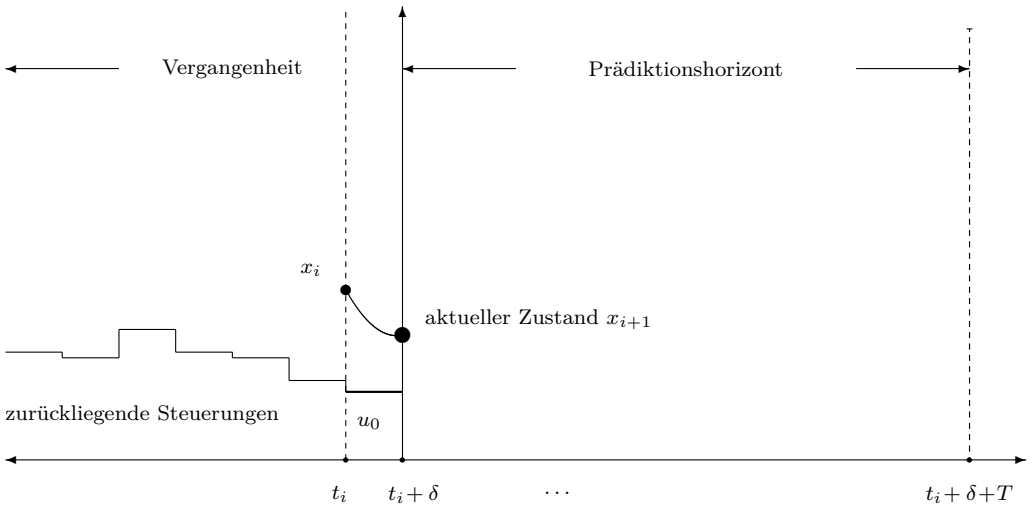

Abbildung 4.3: Ausgangssituation zur Zeit $t_{i+1} = t_i + \delta$

dann beliebig lang fortgesetzt werden. Abbildungen 4.1 bis 4.3 veranschaulichen diese Vorgehensweise.

Aus Gründen der Übersichtlichkeit wurden Rechenzeiten hier vernachlässigt. Aber es wird deutlich, dass eine gewisse Zeit verstreicht, bis aus der Messung des Systemzustands die optimale Steuerung berechnet wird. Außerdem besteht natürlich ein Interesse daran, diese Steuerung schnell berechnen zu können. Ist die Rechenzeit nämlich zu lang, befindet sich das System womöglich beim Anwenden der zurückgelieferten Steuerung mittlerweile in einem anderen Zustand, in dem diese Steuerung dann nicht mehr unbedingt optimal ist.

Löst man die anfallenden Probleme mit der *direkten Mehrzielmethode* im *SQP*-Kontext,[14] schlägt *Diehl* Strategien vor, die die Berechnung beschleunigen können [9, 10, 11]. Wenn schon eine optimale Steuerung für das Intervall $[t_i, t_i + T]$ berechnet und der erste Teil an das System zurückgegeben wurde, kann nach *Diehl* als Startwert $x_{i+1}^{(0)}$ für das neue Optimierungsproblem auf $[t_i+\delta, t_i+\delta+T]$ die Lösung des letzten Problems genutzt werden. Der gemessene Wert $\hat{x}_{i+1}$ dient dabei nicht als Startwert, wird aber über die Nebenbedingung

$$x(t_{i+1}) - \hat{x}_{i+1} = 0 \tag{4.3}$$

in das neue Problem eingebettet. Weiterhin schlägt er vor, auch die optimale Steuerung u als Startwert für die neue diskretisierte Steuerung zu nutzen. So reicht eine einzige *SQP*-Iteration, um die optimale Steuerung auf dem neuen Horizont in linearer Approximation

[14] Siehe Abschnitt 2.2.3.

zu erhalten.

Bei der Wahl der Länge T des *Prädiktionshorizonts* sind zwei gegensätzliche Konsequenzen zu betrachten. Einerseits erfordert die Lösung der Optimierungsprobleme mit zunehmender Länge immer mehr Zeit, so dass T eher klein gewählt werden sollte, andererseits sind die jeweiligen Lösungen immer nur auf dem Prädiktionshorizont $[t_i, t_i + T]$ optimal und ein zu kleiner Prädiktionshorizont kann zu einer Lösung führen, deren Steuerung eine starke Abweichung in der Zielfunktion von der global optimalen Steuerung auf $[t_0, t_f]$ zeigt.

Falls zur Lösung der Optimalsteuerungsprobleme die *Dynamische Programmierung* eingesetzt wird, ist die Wahl von T bezüglich der Laufzeit eher nebensächlich. Wie in Kapitel 2 gesehen, wächst die Laufzeit der *Dynamischen Programmierung* linear mit der Anzahl der Diskretisierungspunkte der Zeit.

Steuert man einen Prozess allerdings *online* – also parallel zum Ablauf – erhält man bei kurzen Horizontlängen auch schnell eine Antwort. Dies kann bei Prozessen, die in zeitlich kleinen Dimensionen ablaufen, sehr von Interesse sein.

Bei Verwendung der *Dynamischen Programmierung* ergibt sich ein weiterer Vorteil. Da beim Durchlauf des Algorithmus für *jeden* Zustandsknoten die optimale Fortsetzung abgespeichert wird, kann – genügend Speicherplatz vorausgesetzt – im Vorfeld des Prozesses das Optimum über einen großen Zeitraum berechnet werden. Falls während der Ausführung des Prozesses dann eine Störung auftritt, so dass die optimale Zustandstrajektorie verlassen wird, kann die optimale Steuerung für den gestörten Zustand in einer Tabelle nachgeschlagen werden. Somit müssen *online* keine Berechnungen mehr durchgeführt werden, allerdings sind geeignete Speicherstrukturen zu empfehlen, um bei großen Datenmengen einen Lesezugriff schnell erfolgen lassen zu können.

Kapitel 5

Modellierung

In dem nun folgenden Kapitel werden wir ein Modell eines LKW vorstellen, mit dem wir die in Kapitel 7 folgenden Berechnungen durchgeführt haben. Außerdem werden wir die Einbindung von Streckendaten in dieses Modell erläutern.

Das in dieser Arbeit behandelte Modell vernachlässigt andere Verkehrsteilnehmer oder sonstige Hindernisse, so dass der LKW sich zu jeder Zeit frei auf der Strecke bewegen kann.

5.1 Modellierung eines LKW

5.1.1 Steuerungen

Die frei wählbaren Steuerungen des LKW sind in Tabelle 5.1 aufgeführt. Die Beschleunigung des Fahrzeugs wird maßgeblich durch das *Motormoment* M_{ind} beeinflusst, die Bremskraft durch das *Bremsmoment* M_{brk}. Aus algorithmischen Gründen steuern wir jeweils deren Änderung R_{ind} bzw. R_{brk}. Der gewählte Gang y stellt eine ganzzahlige Variable dar und bestimmt direkt das *Übersetzungsverhältnis* $i_{\text{T}}(y)$ zwischen Motor und Antriebsachse und den *Wirkungsgrad* $\eta_{\text{T}}(y)$ im Gang y.

Das Gesamt-*Bremsmoment* setzt sich aus zwei Elementen zusammen: zum einen aus dem *Bremsmoment* der *Motorbremse* M_{EB} und zum anderen aus dem des *Retarders* M_{ret}:

$$M_{\text{brk}}(s) := i_{\text{T}}(y)\ M_{\text{EB}}(s) + M_{\text{ret}}(s). \tag{5.1}$$

Wir haben uns dafür entschieden, nur das Gesamtmoment zu steuern. An dieser Stelle

Name	Beschreibung	Einheit	Definitionsbereich
R_{ind}	Änderung des Motormoments	$\frac{\text{Nm}}{\text{s}}$	$[R_{\text{ind,min}}, R_{\text{ind,max}}]$
R_{brk}	Änderung des Bremsmoments	$\frac{\text{Nm}}{\text{s}}$	$[R_{\text{brk,min}}, R_{\text{brk,max}}]$
y	Gewählter Gang	–	$\Omega \subset \mathbb{N}$

Tabelle 5.1: Steuerungen des LKW

Name	Beschreibung	Einheit
A	Frontfläche	m^2
c_w	Strömungswiderstandskoeffizient	–
η_A	Wirkungsgrad der Hinterachse	–
f_r	Rollreibungskoeffizient	–
i_A	Hinterachsübersetzung	–
$n_\mathrm{eng,max}$	Maximale Motordrehzahl	$\frac{1}{\mathsf{min}}$
$n_\mathrm{eng,min}$	Minimale Motordrehzahl	$\frac{1}{\mathsf{min}}$
r_stat	Radius der Hinterreifen	m
η_M	Wirkungsgrad des Motors	–
m	Fahrzeugmasse	kg
$R_\mathrm{ind,max}$	Maximale Zunahme des Motormoments	$\frac{\mathsf{Nm}}{\mathsf{s}}$
$R_\mathrm{ind,min}$	Maximale Abnahme des Motormoments	$\frac{\mathsf{Nm}}{\mathsf{s}}$
$R_\mathrm{brk,max}$	Maximale Zunahme des Bremsmoments	$\frac{\mathsf{Nm}}{\mathsf{s}}$
$R_\mathrm{brk,min}$	Maximale Abnahme des Bremsmoments	$\frac{\mathsf{Nm}}{\mathsf{s}}$
$i_\mathrm{T}(y)$	Übersetzungsverhältnis im Gang y	–
$\eta_\mathrm{T}(y)$	Wirkungsgrad im Gang y	–
g	Gravitationskonstante	$\frac{\mathsf{m}}{\mathsf{s}^2}$
ρ_air	Luftdichte	$\frac{\mathsf{kg}}{\mathsf{m}^3}$

Tabelle 5.2: Feste Parameter und physikalische Konstanten

ergibt sich daher Spielraum für eine unterschiedliche Priorisierung der genannten Bremsarten.

5.1.2 Differentialgleichungssystem

Als unabhängige Variable des Modells haben wir die Position s (in Metern) des LKW auf der Karte gewählt. Daher wird die Anwendbarkeit des Modells auf strikt positive Geschwindigkeiten begrenzt. Die Zeit kann in Abhängigkeit von der – durch die positive Geschwindigkeit *eindeutige* – Position s und der Geschwindigkeit $v(s)$ als differentieller Zustand über die Formel

$$\dot{t}(s) = \frac{1}{v(s)} \tag{5.2}$$

mitgeführt werden, wird aber nicht zwingend benötigt.

Die Geschwindigkeit des Fahrzeugs ergibt sich aus mehreren Komponenten:[15] Dem *Beschleunigungsmoment*

$$M_\mathrm{acc}(s) := i_\mathrm{T}(y)\ \eta_\mathrm{T}(y)\ M_\mathrm{ind}(s), \tag{5.3}$$

[15] Alle im Text nicht erwähnten Größen sind in Tabelle 5.2 zu finden.

dem *Bremsmoment*[16] inklusive Motorreibung in direkter Abhängigkeit der *Motordrehzahl* $n_{\text{eng}}(s)$

$$M_{\text{brake}}(s) := M_{\text{brk}}(s) + i_{\text{T}}(y)\ M_{\text{fric}}(n_{\text{eng}}(s)), \tag{5.4}$$

dem *Strömungswiderstand*

$$M_{\text{air}}(s) := \frac{1}{2}\ c_{\text{w}}\ A\ \rho_{\text{air}}\ v^2(s) \tag{5.5}$$

und dem *Rollwiderstand* in Abhängigkeit von der Steigung des aktuellen Streckenabschnitts $\gamma(s)$,

$$M_{\text{street}}(s) := m \cdot g \cdot (\sin(\gamma(s)) + f_{\text{r}} \cos(\gamma(s)))\,. \tag{5.6}$$

Aus diesen Definitionen ergibt sich die Differentialgleichung für die Geschwindigkeit:

$$\dot{v}(s) = \frac{1}{m \cdot v(s)} \left(\frac{i_{\text{A}}}{r_{\text{stat}}} (M_{\text{acc}}(s) - M_{\text{brake}}(s)) - M_{\text{air}}(s) - M_{\text{street}}(s) \right). \tag{5.7}$$

Die restlichen beiden Zustände sind die bereits angesprochenen Momente: *Motormoment*

$$\dot{M}_{\text{ind}}(s) = \frac{1}{v(s)}\ R_{\text{ind}}(s) \tag{5.8}$$

und *Bremsmoment*

$$\dot{M}_{\text{brk}}(s) = \frac{1}{v(s)}\ R_{\text{brk}}(s). \tag{5.9}$$

Die *Motordrehzahl* aus (5.4) kann aus der aktuellen Geschwindigkeit und dem aktuellen Gang gewonnen werden:

$$n_{\text{eng}}(s) := v(s)\ \frac{i_{\text{A}}\ i_{\text{T}}(y)}{r_{\text{stat}}}\ \frac{60\ [\mathsf{s}]}{2\pi}. \tag{5.10}$$

Die Größen $M_{\text{ret}}(s)$, $M_{\text{EB}}(s)$ und $M_{\text{int}}(s)$ sind durch $M_{\text{ret,max}}$, $M_{\text{EB,max}}$ und $M_{\text{int,max}}$ beschränkt. Dabei sind diese Maximalwerte keine Konstanten im engeren Sinne, sondern Funktionen der *Motordrehzahl*.

5.2 Streckentopografie

Um einen LKW vorausschauend steuern zu können, werden Daten über die vor ihm liegenden Streckenabschnitte benötigt. Daher gehen wir davon aus, dass zu fahrende Strecken vor der Fahrt vermessen wurden oder während der Fahrt die benötigten Daten per Satellitennavigation wie etwa *GPS* oder *Galileo* zur Verfügung stehen.

[16] Man beachte, dass $M_{\text{brake}}(s) \neq M_{\text{brk}}(s)$.

Für unsere Untersuchungen sind – anders als in der Berechnung optimaler Routen – nicht nur die Koordinaten in der Ebene wichtig, sondern z.B. auch die Höhe der Position auf der Straße. Aus den gesammelten Höhendaten lassen sich Steigungsprofile einer Strecke erstellen. Gerade für LKW mit hoher Masse ist die Steigung einer Strecke eine wichtige Information.

Werden die ebenen Koordinaten genau genug gemessen, können außerdem brauchbare Daten über die *Krümmung* einer Kurve gewonnen werden. Diese beeinflussen – wie wir später sehen werden – die Maximalgeschwindigkeit, mit der diese Kurve durchfahren werden kann.

Um die Gesetze des Landes, in dem gefahren wird, nicht zu verletzen, sind darüber hinaus Daten über die jeweils zulässige Höchstgeschwindigkeit unverzichtbar.

5.3 Grenzen und Nebenbedingungen

Zusätzlich zu Bedingungen aus der Fahrzeugphysik existieren in unserem Modell weitere Nebenbedingungen wie etwa die bereits erwähnte gesetzliche Höchstgeschwindigkeit.

Die Steuerungen R_{ind} und R_{brk} unterliegen einfachen Schranken:

$$R_{\text{ind,min}} \leq R_{\text{ind}}(s) \leq R_{\text{ind,max}}, \tag{5.11a}$$

$$R_{\text{brk,min}} \leq R_{\text{brk}}(s) \leq R_{\text{brk,max}}. \tag{5.11b}$$

Je nachdem wie viele Gänge modelliert werden, muss

$$1 \leq y(s) \leq n_g, \quad y(s) \in \mathbb{N} \tag{5.12}$$

gelten. Für jeden modellierten Gang y müssen die Konstanten $i_{\text{T}}(y)$ und $\eta_{\text{T}}(y)$ vorliegen.

Wie bereits erwähnt, werden die verschiedenen Momente drehzahlabhängig begrenzt:

$$0\ [\mathsf{Nm}] \leq\ M_{\text{ind}}(s) \leq M_{\text{ind,max}}(n_{\text{eng}}(s)), \tag{5.13a}$$

$$0\ [\mathsf{Nm}] \leq\ M_{\text{ret}}(s) \leq M_{\text{ret,max}}(n_{\text{eng}}(s)), \tag{5.13b}$$

$$0\ [\mathsf{Nm}] \leq\ M_{\text{EB}}(s) \leq M_{\text{EB,max}}(n_{\text{eng}}(s)). \tag{5.13c}$$

Aus diesen Beschränkungen und (5.1) ergibt sich auch die Grenze für das *Bremsmoment*:

$$0\ [\mathsf{Nm}] \leq\ M_{\text{brk}}(s) \leq i_{\text{T}}(y)\ M_{\text{EB,max}}(n_{\text{eng}}(s)) + M_{\text{ret,max}}(n_{\text{eng}}(s)). \tag{5.14}$$

Wir priorisieren hierbei die Motorbremse. Ist also das gewünschte Bremsmoment allein durch die Motorbremse realisierbar, wird der Retarder nicht eingesetzt. Dieser wird erst zusätzlich benutzt, falls $M_{\text{brk}}(s) > i_{\text{T}}(y)\ M_{\text{EB,max}}(n_{\text{eng}}(s))$.

Die *Motordrehzahl* wird durch vorgegebene Konstanten

$$n_{\text{eng,min}} \leq\ n_{\text{eng}}(s) \leq n_{\text{eng,max}} \tag{5.15}$$

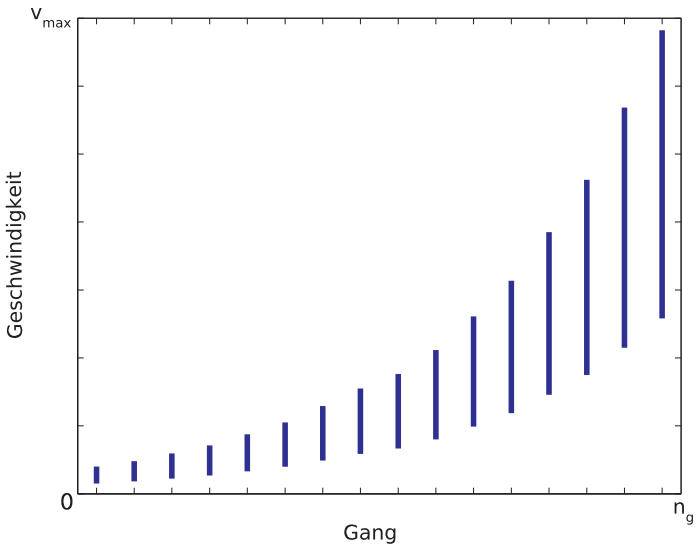

Abbildung 5.1: Schematischer Zusammenhang zwischen wählbaren Gängen und fahrbaren Geschwindigkeiten durch Bedingung (5.15). Gültige Kombinationen sind blau eingefärbt.

des Motors beschränkt. In Verbindung mit (5.10) limitieren diese Grenzen somit den gültigen Geschwindigkeitsbereich, der in Gang y gefahren werden darf. Der Zusammenhang wird in Abbildung 5.1 verdeutlicht.

Zusätzlich existieren Grenzen für die Geschwindigkeit, die einerseits von der *Krümmung* $\kappa(s)$ der Strecke an der aktuellen Position s und andererseits von gesetzlichen Bestimmungen abhängen:

$$v(s) \leq v_{\text{curve}}\big(\kappa(s)\big), \tag{5.16a}$$
$$v(s) \leq v_{\text{law}}(s). \tag{5.16b}$$

Diese Bedingungen werden zu

$$v(s) \leq \min(v_{\text{curve}}(\kappa(s)), v_{\text{law}}(s)) \tag{5.17}$$

zusammengefasst.

Als sinnvoll wird allgemein die Komplementarität der zu steuernden Momente erachtet, denn unabhängig von der Streckenposition sollte niemals gleichzeitig beschleunigt und gebremst werden, daher gilt:

$$(M_{\text{ind}}(s) \geq 0 \wedge M_{\text{brk}}(s) = 0) \vee (M_{\text{ind}}(s) = 0 \wedge M_{\text{brk}}(s) \geq 0)\,. \tag{5.18}$$

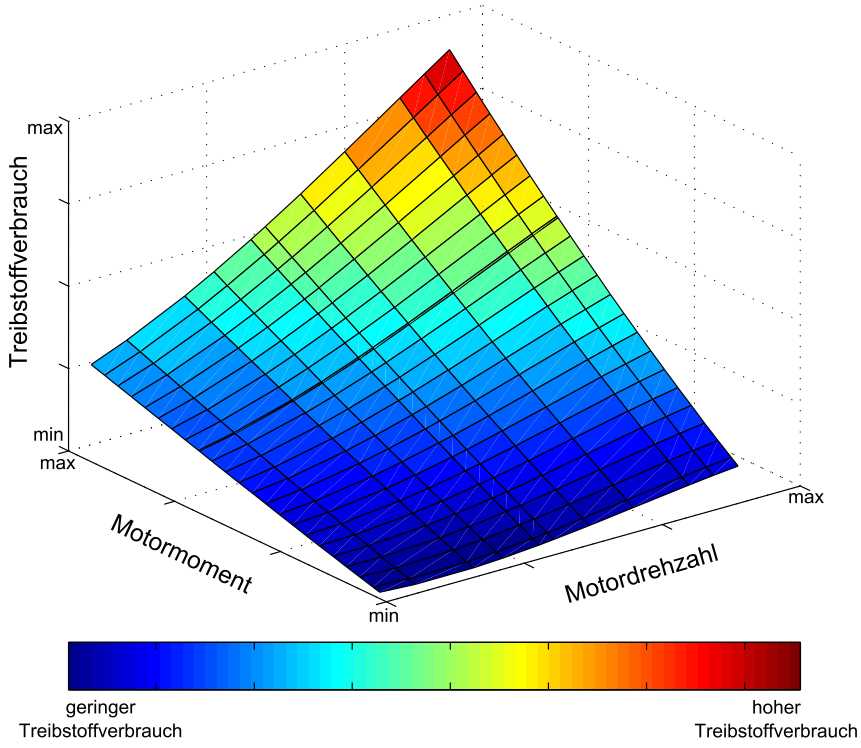

Abbildung 5.2: Treibstoffverbrauch in Abhängigkeit von Motormoment und -drehzahl.

5.4 Zielfunktion

Wir haben uns dafür entschieden, verschiedene Faktoren in die Zielfunktion einfließen zu lassen. Die Zielfunktion ist somit eine Linearkombination verschiedener Terme

$$\Phi := \sum_{i=0}^{k} a_i \cdot J_i \tag{5.19}$$

mit frei wählbaren Gewichten $a_i \in \mathbb{R}$.

In Zeiten steigender Ressourcenknappheit ist sowohl aus ökologischer als auch ökonomischer Sicht die Minimierung des Treibstoffverbrauchs eines Fahrzeuges zunehmend interessant. Den Treibstoffverbrauch Q_{fuel} haben wir als Funktion des *Motormoments* und der *Motordrehzahl* modelliert, siehe Abbildung 5.2. Der Treibstoffverbrauch über

eine Strecke $[0, s_{\text{horiz}}]$ ist somit

$$J_0 = J_{\text{fuel}} := \int_0^{s_{\text{horiz}}} Q_{\text{fuel}}(M_{\text{ind}}(s), n_{\text{eng}}(s))\ ds. \tag{5.20}$$

Da wir die Fahrtzeit nicht direkt beschränken, bestrafen wir weiterhin die Unterschreitung einer Wunschgeschwindigkeit, die als von der *Krümmung* und *Steigung* abhängige Funktion $v_{\text{desire}}(s)$ modelliert ist. Durch Maximumsbildung wird sichergestellt, dass eine Überschreitung hier nicht bestraft wird.[17] Wir haben uns für eine quadratische Bestrafung entschieden, was dazu führt, dass hohe Abweichungen von der Wunschgeschwindigkeit stärker bestraft werden, als es bei einem linearen Term der Fall wäre, während kleine Abweichung toleriert werden:

$$J_1 = J_{\text{desire}} := \int_0^{s_{\text{horiz}}} \max(v_{\text{desire}}(s) - v(s), 0)^2\ ds. \tag{5.21}$$

Schnelle Änderungen des *Motormoments* M_{ind} und des *Bremsmoments* M_{brk} werden als unerwünscht betrachtet, da diese den Komfort des Fahrers negativ beeinflussen. Deswegen werden auch die Änderungen dieser Größen in Abhängigkeit von der aktuellen Geschwindigkeit quadratisch bestraft:

$$J_2 = J_{\text{comfort}} := \int_0^{s_{\text{horiz}}} \frac{1}{v^2(s)} \left(R_{\text{ind}}(s)^2 + R_{\text{brk}}(s)^2\right) ds. \tag{5.22}$$

5.4.1 Schaltkosten

In diesem Modell wird angenommen, dass der Wechsel eines Gangs instantan erfolgen kann. Daher kann es dazu kommen, dass in etwaigen Lösungen der Gang sehr häufig gewechselt wird. Um diesem Phänomen entgegenzusteuern, existieren verschiedene Strategien:

- Die naheliegendste Möglichkeit ist wohl, einen konstanten Strafterm für einen Gangwechsel zur Zielfunktion zu addieren. Aufgrund der Unstetigkeit liefert diese Vorgehensweise aber Probleme, wenn das Optimalsteuerungsproblem mit einem ableitungsbasierten Verfahren – wie in Abschnitt 2.2 vorgestellt – gelöst werden soll.
- Ein anderer Weg ist, die Anzahl der Schaltvorgänge auf einer Strecke einer vorgegebenen Länge L durch eine Konstante M zu beschränken. Diese Art von Beschränkung ist wiederum ungeeignet für die Verwendung der *Dynamischen Programmierung*, da diese sequentiell abläuft und darauf basiert, dass bereits berechnete Endstücke optimal sind. Wurde auf einem Endstück mit einer Länge $l < L$

[17] Eine erhöhte Geschwindigkeit wird bereits durch den erhöhten Treibstoffverbrauch indirekt bestraft.

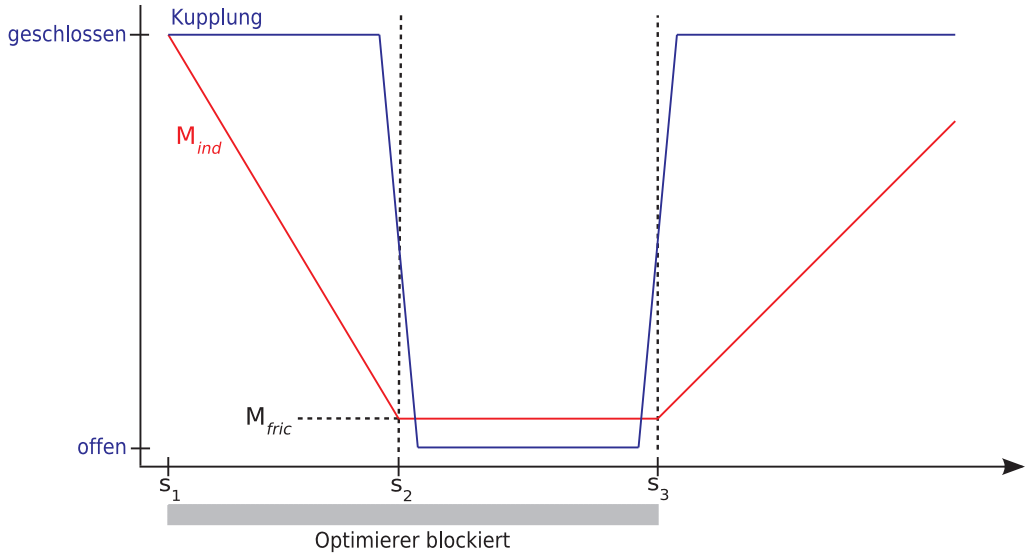

Abbildung 5.3: Schematische Darstellung eines Modells der für einen Gangwechsel benötigten Transitionsstufe: Zum Einleiten des Gangwechsels muss das Motormoment auf das Niveau der Motorreibung reduziert werden. Danach erfolgt die Öffnung der Kupplung und die Phase, in der der Gang gewechselt wird. Nach Schließung der Kupplung kann das Motormoment wieder frei erhöht werden.

eine Optimalsteuerung mit M Schaltvorgängen berechnet, wird zwangsläufig auf dem $L-l$ langen Stück davor nicht geschaltet. Es werden also manche Mengen des Suchraums abgeschnitten, so dass das wahre Optimum übersehen werden kann.

- Es besteht außerdem die Möglichkeit, die Geschwindigkeitsänderung, die während eines Gangwechsels auftritt, über ein genaues Modell des Schaltvorgangs zu simulieren. In diesem Modell müssten dann u.a. Zeiten für das Ein- und Auskuppeln berücksichtigt werden. Durch eine Vielzahl an Simulationen kann versucht werden, die Geschwindigkeitsänderung als Funktion von Eingangsgrößen wie Gang, Steigung, Motor- bzw. Bremsmoment und Geschwindigkeit darzustellen. Diese Funktion kann dann in die Dynamik des ursprünglichen Modells einbezogen werden, um den Geschwindigkeitsverlust bzw. -gewinn mitzusimulieren und zu optimieren.

Wir entscheiden uns aufgrund der genannten Probleme für die erste Variante und definieren

$$J_3 = \begin{cases} 1 & \text{falls Gangwechsel erfolgt} \\ 0 & \text{sonst} \end{cases} . \tag{5.23}$$

Bei der Wahl der Gewichte a_i müssen die Größenordnungen der verschiedenen Zielfunktionen J_i berücksichtigt werden. Der Treibstoffverbrauch in Litern oder Kilogramm ist beispielsweise wesentlich geringer als quadratische Änderungen der Momente auf einer Strecke gleicher Länge. Eine beispielhafte Wahl könnte in folgender Größenordnung erfolgen:

$$a_0 = 0.1;\ a_1 = 1.0;\ a_2 = 0.01;\ a_3 = 0.5 \tag{5.24}$$

Den Einfluss der Wahl der Gewichte werden wir in Kapitel 7 noch einmal aufgreifen.

Kapitel 6

TruckDP

In diesem Kapitel soll nun die im Rahmen dieser Arbeit entstandene Software *TruckDP* vorgestellt werden. Sie verwendet die in Kapitel 2 vorgestellte *Dynamische Programmierung*, um einen in Kapitel 5 modellierten LKW optimal zu steuern.

6.1 Implementierung

6.1.1 Diskretisierung

Wir betrachten zunächst ein Optimierungsproblem auf festem Horizont $[s_0, s_N]$. Die Strecke wird zur Diskretisierung in Teilstücke der Länge δ_s unterteilt. Um Berechnungen zu vereinfachen, wählen wir δ_s so, dass

$$\delta_s \mid (s_f - s_0) \tag{6.1}$$

gilt. Wie wir in Kapitel 2 gesehen haben, muss bei der *Dynamischen Programmierung* eine Diskretisierung der reellwertigen Zustände erfolgen. Die differentiellen Zustände Geschwindigkeit (v) und Motor- bzw. Bremsmoment (M_{ind} bzw. M_{brk}) werden daher ebenfalls diskretisiert behandelt. Da sich die Größen M_{ind} und M_{brk} durch Bedingung (5.18) derart beeinflussen, dass nicht beide gleichzeitig größer als 0 sein können, findet keine getrennte Betrachtung statt. Die diskretisierten Werte der Momente werden in einem gemeinsamen Vektor, der den Bereich $[-M_{\text{brk,max}}, M_{\text{ind,max}}]$ abbildet, gespeichert. Das Bremsmoment wird also im negativen und das Motormoment im positiven Bereich abgespeichert, so dass die einzelnen Momente über

$$(M_{\text{ind,k}}, M_{\text{brk,k}}) = \begin{cases} (M_k, 0) & \text{falls } M_k \geq 0 \\ (0, -M_k) & \text{sonst} \end{cases} \tag{6.2}$$

zurückgewonnen werden können. Die Feinheit der jeweiligen Diskretisierung wird der Software mit δ_v und δ_M übergeben. Die Gänge $(1, \ldots, n_g)$ liegen bereits diskret vor.

Als Steuerungen sind die Änderungen von Gang (Δy) und Brems- bzw. Motormoment (ΔM) implementiert. Die Änderung der Momente ist ein ganzzahliges Vielfaches von δ_M.

Die Übergangsfunktionen $x_{k+1} = f(x_k, u_k)$, die benötigt werden, um die Zustände an der Stelle $(k+1) \cdot \delta_s$ zu erhalten, sind somit

$$y_{k+1} = y_k + \Delta y_k, \tag{6.3a}$$
$$M_{k+1} = M_k + \Delta M_k. \tag{6.3b}$$

Um die neue Geschwindigkeit v_{k+1} zu erhalten, wird die Differentialgleichung (5.7) mit einem numerischen Integrator für den Abschnitt der Länge δ_s gelöst. Das reellwertige Ergebnis der Integration wird anschließend auf den naheliegendsten Diskretisierungspunkt der Geschwindigkeit gerundet. Als Integrator benutzen wir ein *Runge-Kutta-Verfahren* vierter Ordnung mit variabler Schrittweite.

Die Steuerungen ΔM und Δy werden direkt bei $s_k = s_0 + k \cdot \delta_s$ an das Modell übertragen, so dass auf dem Intervall (s_k, s_{k+1}) bereits die neuen Momente und der neue Gang gelten. Das Ergebnis des *Anfangswertproblems*, das gelöst werden muss, um die neue Geschwindigkeit zu erhalten, hängt hingegen maßgeblich von den gerade erwähnten Größen auf dem Integrationshorizont $[s_k, s_{k+1}]$ ab. Daher haben wir uns dazu entschieden, die Änderung der Geschwindigkeit bei s_{k+1} zu realisieren.

Die Kostenfunktion $g_k(x_k, u_k)$ der auf dem k-ten Streckenabschnitt entstehenden Kosten wählen wir – wie in Kapitel 5 diskutiert – als

$$J_{\text{total}} = \sum_{i=0}^{3} a_i J_i \tag{6.4}$$

mit

$$J_0 = Q_{\text{fuel}}(M_{\text{ind,k}}, n_{\text{eng,k}}), \tag{6.5a}$$
$$J_1 = \max(v_{\text{desire,k}} - v_k, 0)^2, \tag{6.5b}$$
$$J_2 = \frac{\Delta M_{\text{ind,k}}^2 + \Delta M_{\text{brk,k}}^2}{v_k^2}, \tag{6.5c}$$
$$J_3 = \begin{cases} 1 & \text{falls } y_k \neq y_{k+1} \\ 0 & \text{sonst} \end{cases} \tag{6.5d}$$

und frei einstellbaren Gewichten $a_i \in \mathbb{R}$.

6.1.2 Beschleunigung des Algorithmus

Bevor wir im nächsten Abschnitt den Algorithmus explizit angeben, wollen wir zunächst den Ablauf skizzieren und an beachtenswerten Stellen unsere Vorgehensweise erläutern.

Der Algorithmus startet bei Streckenposition $s = s_{N-1}$. Es müssen für jede Kombination an Zuständen

$$(v_{N-1}, M_{N-1}, y_{N-1}) \tag{6.6}$$

die Fortsetzungskosten und die optimalen Steuerungen berechnet werden. Um den *Suchraum* zu verkleinern, werden an dieser Stelle direkt diejenigen Kombinationen verworfen, die mit Hilfe der Beschränkungen (5.15) und der Formel für die Berechnung der *Motordrehzahl* (5.10) identifiziert werden können. Durch das frühe Ausschließen dieser Zustände werden *alle* Integrationen von diesem Zustand aus eingespart. Ungültige Gang-Geschwindigkeit-Kombinationen erhalten eine Markierung, so dass man sie später wiedererkennen kann.

Für jedes gültige Zustandstripel muss eine Minimierung der Kostenfunktion (6.4) über alle *möglichen* Steuerungen durchgeführt werden. Pro Steuerung wird u.a. ein *Anfangswertproblem* zur Bestimmung der neuen Geschwindigkeit v_N gelöst. Auch hier versucht man Steuerungen über Nebenbedingungen an die neuen Zustände auszuschließen, um Auswertungen der Kostenfunktion zu sparen. Falls die einfachen Grenzen (5.11) erfüllt werden, muss allerdings die Integration der Geschwindigkeit durchgeführt werden.

Ist die neue Geschwindigkeit v_N gültig – sind also alle Nebenbedingungen erfüllt – kann es trotzdem vorkommen, dass es keine gültige Fortsetzung vom Zustandsknoten $x_{N-1} = (v_{N-1}, M_{N-1}, y_{N-1})$ aus gibt. Dies kann beispielsweise eintreten, wenn v_{N-1} in der Nähe der vorgegebenen Minimalgeschwindigkeit v_{low} liegt und auf dem nächsten Streckenabschnitt eine hohe Steigung γ folgt, so dass *alle* Steuerungen zu einer Unterschreitung von v_{low} führen. Für den Knoten x_{N-1} muss also vermerkt werden, dass es keine Fortsetzung gibt, da Steuerungen, die von einem Zustand x_N aus zu x_{N-1} führen, dementsprechend auch nicht als gültig angesehen werden dürfen.

Da in das Anfangswertproblem zur Berechnung von v_N die zum Zeitpunkt $s = s_{N-1}$ geschalteten neuen Zustände M_N und y_N eingehen, treten an mehreren Stellen innerhalb der Schleife über die Zustände die exakt gleichen Aufrufe des Integrators auf. Dies passiert genau dann, wenn von unterschiedlichen Zuständen in die gleichen neuen Zustände geschaltet wird. Um den Rechenaufwand an dieser Stelle zu verkleinern, werden die Integratorergebnisse abgespeichert, um sie beim nächsten Aufruf lediglich in einer Tabelle nachschlagen zu müssen.

Weiterhin lassen sich über Monotonieeigenschaften gleich mehrere Steuerungen ausschließen. Falls man von Zustand $x_{N-1} = (v_{N-1}, M_{N-1}, y_{N-1})$ aus mit einer Steuerung $u_{N-1} = (\Delta M_{N-1}, \Delta y_{N-1})$ fortfährt und sich aus dieser Steuerung beispielsweise eine Geschwindigkeit $v_N > v_{max,N}$ ergibt, brauchen die Steuerungen mit einer Momentenänderung $\Delta M > \Delta M_{N-1}$ nicht mehr betrachtet zu werden. Gleiches gilt auch umgekehrt für Unterschreitungen einer dem Algorithmus vorgegebenen Minimalgeschwindigkeit.

Wurden so für alle gültigen Zustandskombinationen die optimalen Steuerungen mit den optimalen Fortsetzungskosten berechnet, fährt man bei $s = s_{N-2}$ mit dem nächsten Schleifendurchlauf fort.

6.1.3 Algorithmus

Algorithmus 2. (TruckDP)

- *Eingabe: Startposition s_0, Endposition s_f, Auflösung δ_s der örtlichen Diskretisierung; Grenzen für Geschwindigkeit (v_{low}, v_{up}) und Momente (M_{low}, M_{up}) und die jeweilige Diskretisierungsauflösung δ_v bzw. δ_M. Weiterhin die Streckendaten und die Gewichte a_i der Zielfunktion. Optional: Anfangswerte der Zustände.*
- *Schleife über $k = N-1, \ldots, 0$*
  1. *Lese Streckendaten (Steigung γ, Maximalgeschwindigkeit v_{max}) zu $s = s_0 + k \cdot \delta_s$ ein.*
  2. *Schleife über alle Zustandskombinationen (v_k, M_k, y_k)*
     a) *Falls $v_k > v_{max}$, überspringe aktuelle Kombination und schließe alle Kombinationen mit $v > v_k$ aus.*
     b) *Prüfe anhand von (5.10) und (5.15), ob Gang und Geschwindigkeit eine gültige Drehzahl liefern. Falls $n_{\text{eng}} > n_{\text{eng,max}}$, verwerfe alle Kombinationen mit höherer Geschwindigkeit bei gleichem Gang und niedrigerem Gang bei gleicher Geschwindigkeit. Verfahre analog dazu, falls $n_{\text{eng}} < n_{\text{eng,min}}$.*
     c) *Schleife über alle Steuerungen $(\Delta M_k, \Delta y_k)$*
        i. *Berechne $M_{k+1} = M_k + \Delta M_k$ und $y_{k+1} = y_k + \Delta y_k$.*
        ii. *Prüfe analog zu Schritt 2b), ob y_{k+1} und v_k gültige Drehzahl ergeben.*
        iii. *Berechne v_{k+1} durch Integration und anschließendes Runden (oder schlage es nach, falls in einer vorherigen Iteration bereits berechnet) und überprüfe wie in Schritten 2a) und 2b) die neue Geschwindigkeit auf Gültigkeit.*
        iv. *Schlage Fortsetzungskosten zu $x_{k+1} = (v_{k+1}, M_{k+1}, y_{k+1})$ nach. Falls nicht vorhanden, da x_{k+1} keine gültige Fortsetzung besitzt, verwerfe aktuelle Steuerung.*
        v. *Berechne Kosten $g_k(x_k, u_k)$ und den daraus resultierenden Zielfunktionswert $J(x_k) = g_k(x_k, u_k) + J(x_{k+1})$.*
        vi. *Falls Zielfunktionswert kleiner als aktuell bekannter am Knoten x_k, ersetze das bisherige Minimum durch den neuen Wert und speichere die aktuelle Steuerung als optimale Fortsetzung zu x_k ab.*
  3. *$k = k - 1$*
- *Bestimmung der optimalen Trajektorie: Suche Zustand x_0^* mit minimalem Zielfunktionswert und schlage zugehörige optimale Steuerung u_0^* nach.*

- *Schleife über* $k = 0, \ldots, N-1$
  1. *Berechne nächsten Zustand der optimalen Trajektorie* $x^*_{k+1} = f(x^*_k, u^*_k)$.
  2. *Schlage* u^*_{k+1} *nach.*

6.1.4 Modifikationen für Systeme mit wenig Arbeitsspeicher

In der oben beschriebenen Variante müssen mitunter große Mengen an Daten im Speicher gehalten werden. Besteht die Schwierigkeit, dass für das zu lösende Problem nicht genug Arbeitsspeicher zur Verfügung steht, verlangsamt die vom Betriebssystem gesteuerte Auslagerung der Daten auf die Festplatte den Algorithmus erheblich, da dem Betriebssystem nicht bekannt ist, welche Daten wann gebraucht werden. Um dieses Problem zu umgehen, können temporär nicht benötigte Daten aber gezielt auf die Festplatte ausgelagert werden.

Beim i-ten Durchlauf der äußeren Schleife über k ($k = i$) müssen lediglich die Fortsetzungskosten zu allen Zustandsknoten aus dem vorherigen Durchlauf $k = i + 1$ für die Bestimmung der aktuellen Zielfunktionswerte bekannt sein. Die Fortsetzungskosten für $k \geq i + 2$ werden zu diesem Zeitpunkt gar nicht mehr benötigt und die optimalen Steuerungen für $k \geq i + 1$ erst wieder, wenn die optimale Trajektorie vorwärts berechnet wird. Erstere können somit gelöscht, letztere bei Bedarf zeitweise auf die Festplatte ausgelagert werden.

Praktisch gesehen kann die Länge des Horizonts daher temporär auf ein beliebiges ganzzahliges Vielfaches von δ_s reduziert werden. Die Berechnungen müssen allerdings – im Unterschied zum *Gleitenden Horizont* – weiterhin von hinten nach vorne durchgeführt werden. Es werden also zuerst die Fortsetzungskosten auf den Endknoten berechnet und wenn sie dann aus dem Horizont herausfallen, müssen die Kosten in die Funktionen g_N integriert werden. Erst bei der Vorwärts-Simulation müssen die Daten wieder nach und nach in den Speicher geladen werden. Im folgenden Algorithmus ist die genaue Vorgehensweise beschrieben.

Algorithmus 3. (TruckDP – Splitting)

1. *Eingabe: Die gleichen Werte wie in Algorithmus 2.*
2. *Bestimme – abhängig von der Größe des zur Verfügung stehenden Arbeitsspeichers – die Länge* L *eines Teilhorizonts, so dass* $n \cdot L = (s_f - s_0)$ *für* $n \in \mathbb{N}$.
3. *Schleife über* $i = n, \ldots, 1$
   a) $s_a = s_0 + (i-1) \cdot L, \quad s_b = s_0 + i \cdot L$
   b) *Falls* $i < n$*: Initialisiere die Endkosten bei* s_b *mit den im Arbeitsspeicher gehaltenen Fortsetzungskosten.*
   c) *Wende Algorithmus 2 auf* $[s_a, s_b]$ *an.*

   d) *Speichere die optimalen Fortsetzungssteuerungen für alle Zustandsknoten auf der Festplatte.*
   e) *Lösche die gerade gespeicherten Daten aus dem Arbeitsspeicher und verwerfe zusätzlich alle Fortsetzungskosten, abgesehen von denen der Zustände bei s_a.*

4. *Bestimmung der optimalen Trajektorie: Suche Zustand x_0^* mit minimalem Zielfunktionswert und schlage zugehörige optimale Steuerung u_0^* nach.*
5. *Schleife über $k = 0, \ldots, N-1$*
   a) *Berechne nächsten Zustand der optimalen Trajektorie $x_{k+1}^* = f(x_k^*, u_k^*)$.*
   b) *Falls $k \cdot \delta_s = c \cdot L$ für ein $c \in \mathbb{N}$: Lösche optimale Steuerungen aus dem Speicher und ersetze sie durch die Daten des nächsten Horizonts der Länge L von der Festplatte.*
   c) *Schlage u_{k+1}^* nach.*

Theorem 4 (Vergleich der Lösungen)**.** *Die beiden Algorithmen 2 und 3 liefern exakt das gleiche Ergebnis, also die gleiche Optimalsteuerung mit gleichem Zielfunktionswert und ebenfalls die gleichen optimalen Fortsetzungen im Falle einer Störung.*

Beweis. Wir betrachten hier der Einfachheit halber nur den Fall für $n = 2$ (Konstante aus Algorithmus 3, Schritt 2), da lediglich die Tatsache, *dass* es einen Übergang mit Datenverlagerung gibt, von Bedeutung ist. Die Anzahl der Übergänge ist unerheblich, da sich die Übergänge allesamt gleich verhalten. Weiterhin gehen wir davon aus, dass L als ganzzahliges Vielfaches von δ_s gewählt wurde.

Sehen wir uns zuerst die Iteration $i = n = 2$ der Schleife 3 an. Es wird direkt Algorithmus 2 aufgerufen, es treten daher vorerst keinerlei Unterschiede auf.

Die interessante Stelle ist beim Übergang zu $n = 1$ gegeben. Die optimalen Steuerungen werden lediglich auf die Festplatte verlagert, während der Großteil der Zielfunktionswerte verworfen wird. Das Ende des neuen Intervalls $[s_0, s_0 + L]$ ist der Anfang des alten. Die Endkosten für alle Zustände bei $s = s_0 + L$ werden also mit den ursprünglich jeweils eigenen Kosten reinitialisiert. Daher entstehen bei Berechnung der Zielfunktionswerte und optimalen Steuerungen für $s = s_0 + L - \delta_s$ in beiden Algorithmen die gleichen Werte. Daraus folgt dann direkt die Gleichheit aller Werte bis einschließlich $s = s_0$.

Die Rekonstruierung der optimalen Trajektorie verläuft in beiden Algorithmen auf die gleiche Weise. In Algorithmus 3 sind die benötigten Daten der optimalen Fortsetzungen lediglich auf die Festplatte ausgelagert worden. Diese werden sequentiell eingelesen, weshalb sich auch die gleiche Lösungstrajektorie ergibt. □

6.2 Erweiterung: Gleitender Horizont

Das Konzept des *Gleitenden Horizonts* wurde bereits in Kapitel 4 erläutert. Wir wollen in diesem Abschnitt erläutern, wo die Stärken dieses Ansatzes liegen und welche

Anpassungen der Software *TruckDP* wir vorgenommen haben, um mit dieser Technik Lösungen berechnen zu können.

6.2.1 Anwendungsgebiet

Das Hauptanwendungsgebiet des folgenden Algorithmus ist der Online-Kontext, d.h. die optimale Steuerung wird parallel zur Fahrt bestimmt. Er kann eingesetzt werden, wenn nicht sämtliche Daten des Horizonts $[s_0, s_f]$ bei Fahrtantritt verfügbar oder bekannt sind, z.B. bei unbekannter Strecke oder einem ungeplanten Streckenwechsel aufgrund von Straßensperrungen oder Stau. Bei Modellerweiterungen, die auch andere Verkehrsteilnehmer berücksichtigen, ist diese Methode unumgänglich.

Zu Anfang des Algorithmus müssen die Daten lediglich für den Prädiktionshorizont der Länge L bekannt sein. Die restlichen Informationen müssen während der Fahrt – z.B. per Satellitennavigation – nachgeladen werden.

Es sollte bei Verwendung von Algorithmus 4 allerdings beachtet werden, dass die *Dynamische Programmierung* hohe Laufzeiten verursacht und aus diesem Grunde nur für relativ kleine Modelle – wie dem in Kapitel 5 vorgestellten – *online* sinnvoll einsetzbar ist. Will man das Modell auf zusätzliche Zustände oder Steuerungen erweitern, sollte man sich des exponentiellen Wachstums der Laufzeit der *Dynamischen Programmierung* mit der Anzahl der Zustände und Steuerungen bewusst sein. Dadurch sind der *Dynamischen Programmierung* im Online-Kontext enge Grenzen gesetzt.

Weitere Anwendung kann diese Variante von Algorithmus 2 – abgesehen vom Online-Kontext – z.B. dann finden, wenn sowohl Speicherplatz im Arbeitsspeicher als auch auf der Festplatte (oder anderen geeigneten Massenspeichern) knapp ist. Im Gegensatz zu Algorithmus 2 müssen und können die Fortsetzungskosten und optimalen Steuerungen nur für die Zustandsknoten gespeichert werden, die sich in $[s_k, s_k + T]$ befinden. Nach jedem Schleifendurchlauf 3, d.h. wenn die optimale Fortsetzung zu x_k^* berechnet wurde, *müssen* Fortsetzungskosten und optimale Fortsetzungen gelöscht werden. Im um δ_s verschobenen Horizont werden erstmalig die Informationen aus dem Abschnitt $[s_k + T, s_{k+1} + T]$ verarbeitet, die dazu führen, dass Fortsetzungskosten und optimale Fortsetzungen aus dem vorherigen Schleifendurchlauf nicht mehr aktuell sind.

Ein weiterer Grund für die Implementierung dieses Algorithmus ist die Möglichkeit, Analysen und Vergleiche mit Algorithmus 2 zu erzeugen, siehe Abschnitt 7.2.

6.2.2 Anpassung

Die Länge T des Prädiktionshorizonts wird dem Algorithmus vorgegeben. Nun wird zunächst auf dem Horizont $[s_0, s_0 + T]$ durch Anwenden von Algorithmus 2 die optimale Steuerung berechnet. Diese wird allerdings nicht vollständig, sondern lediglich ihr erster Teil u_0^* auf das System aufgetragen. Daraus resultiert der Zustand x_1^*. Das nächste Optimalsteuerungsproblem wird auf dem Horizont $[s_0 + \delta_s, s_0 + \delta_s + T] = [s_1, s_1 + T]$

mit festem Anfangswert x_1^* gelöst. Alle Berechnungen, die bei diesem Durchlauf anfallen und auf dem Intervall $[s_1, s_0 + T]$ auftreten, müssen nicht erneut berechnet werden. Die Ergebnisse teurer Berechnungen – wie z.B. die der Integration der Differentialgleichung für die Geschwindigkeit – können beim ersten Aufruf von Algorithmus 2 abgespeichert werden.

Durch die Wiederverwendung der Ergebnisse und die lineare Laufzeit der *Dynamischen Programmierung* in der Länge des Horizonts, auf dem die Optimalsteuerung berechnet werden soll, ergibt sich somit auf einem Horizont $[a, b]$ die gleiche Laufzeit, die auch ohne die Erweiterung *Gleitender Horizont* entsteht.

6.2.3 Algorithmus

Algorithmus 4. (TruckDP: Erweiterung *Gleitender Horizont*)

1. *Eingabe: Die gleichen Werte wie in Algorithmus 2. Zusätzlich die Länge T des Prädiktionshorizonts.*
2. *Berechne Anzahl der Horizonte* $l := \frac{(s_f - s_0) - T}{\delta_s} + 1$.
3. *Schleife über* $k = 0, \ldots, l$
   a) *Rufe Algorithmus 2 für den Horizont* $[s_k, s_k + T]$ *mit Anfangswert* x_k^* *auf.*[18]
   b) *Berechne* $x_{k+1}^* = f(x_k^*, u_k^*)$.
   c) $k = k + 1$
4. *Für den Abschnitt* $[s_l, s_f] = [s_l, s_l + T]$ *werden sämtliche Punkte der auf diesem Abschnitt optimalen Trajektorie in den Vektor* x^* *übernommen.*

Wenn man also $T = s_f - s_0$ wählt, ist das Ergebnis das gleiche wie das von Algorithmus 2, da keine weiteren Berechnungen als der einmalige Aufruf von Algorithmus 2 erfolgen. Dieses Ergebnis stellt auch das globale Optimum für den Horizont $[s_0, s_f]$ dar. Verkleinert man T für Algorithmus 4, kann diese globale Optimalität nicht mehr garantiert werden. Bei sich relativ gering ändernden Streckendaten wie Steigung, Wunsch- und Maximalgeschwindigkeit ist allerdings mit guten Ergebnissen zu rechnen, da die optimalen Steuerungen auf den jeweils ersten Abschnitten eines Prädiktionshorizonts mit denen der globalen Optima weitgehend übereinstimmen sollten. Treten dagegen Veränderungen der Strecke – wie eine starke Steigung – auf, die sich (vorerst) nicht innerhalb des Prädiktionshorizonts befinden, werden diese nicht berücksichtigt und es ist mit global gesehenen suboptimalen Steuerungen zu rechnen. Eine genauere Analyse dieses Sachverhalts erfolgt in Abschnitt 7.2.

[18] Falls bei s_0 kein Anfangswert gefordert ist, wird das Problem bei $k = 0$ mit freiem Anfangswert gelöst.

Kapitel 7

Ergebnisse und Auswertung

Nachdem wir im vorherigen Kapitel die Software *TruckDP* vorgestellt haben, werden wir in diesem Kapitel die Ergebnisse präsentieren, die mit diesem Algorithmus berechnet wurden. Dabei wollen wir zunächst Optimallösungen vorstellen, bevor wir diese dann in Relation zu anderen Lösungen betrachten.

Wir haben ein Modell mit 16 Gängen gewählt, die in den Plots von 0 bis 15 durchnummeriert sind. In allen folgenden Plots der Geschwindigkeit ist die Wunschgeschwindigkeit blau gestrichelt, die Maximalgeschwindigkeit grün gestrichelt und die optimale Geschwindigkeitstrajektorie rot dargestellt.

7.1 Bestimmung globaler Optima

7.1.1 Strecke mit konstanter Maximalgeschwindigkeit

Zuerst wollen wir Ergebnisse vorstellen, die auf einer Strecke fester Länge berechnet wurden. In Abbildung 7.1 sind die Ergebnisse einer 100 km langen – und auf echten Daten einer deutschen Autobahn basierenden – Strecke dargestellt. Wie man am Graphen der Steigung[19] sieht, treten mehrere lange Streckenabschnitte mit großen Steigungen auf, bei denen der LKW die Wunschgeschwindigkeit v_{desire} nicht halten kann.

Davon abgesehen zeichnet sich die Lösung vor allem durch wenige Schaltvorgänge aus, lediglich an den eben angesprochenen steilen Anstiegen wird heruntergeschaltet, um ein möglichst großes Beschleunigungsmoment zu erreichen und so schnell wie möglich wieder die Wunschgeschwindigkeit zu erlangen. Zu bemerken ist hier, dass nur selten im höchsten Gang gefahren wird, was den Treibstoffverbrauch senken würde. Dies kann an den von uns als Standard gewählten Zielfunktionsgewichten $a_0 = 0.1$, $a_1 = 1.0$, $a_2 = 0.01$, $a_3 = 0.5$ liegen, vergleiche im Gegensatz dazu Abbildung 7.10, bei der $a_3 = 0$ gewählt wurde.

[19] Man beachte, dass nicht die Höheninformationen, sondern deren Änderungen – die Steigung – dargestellt sind.

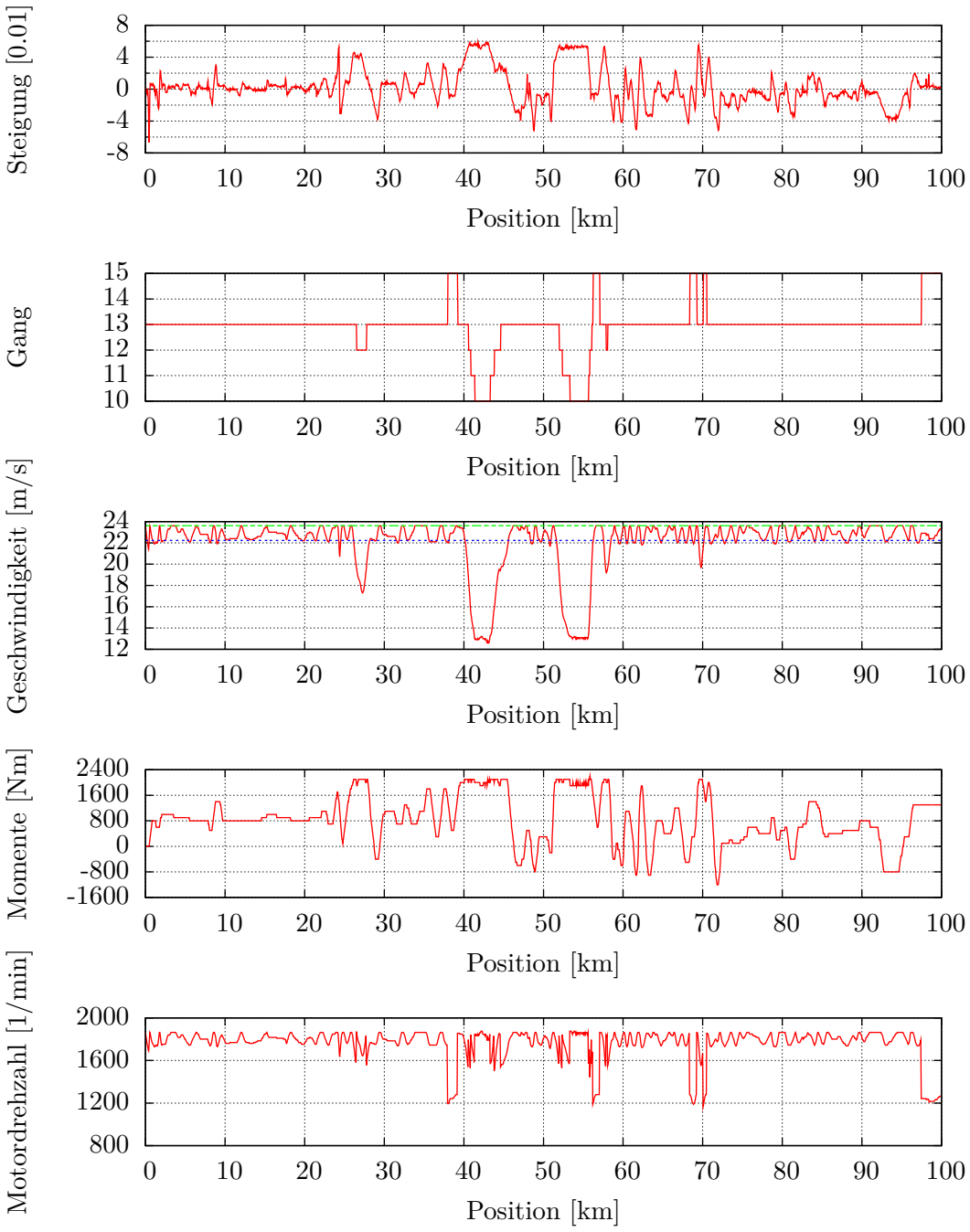

Abbildung 7.1: Ergebnisse auf einer Strecke von 100 km Länge. Die Rechnung erfolgte mit $\delta_s = 50$ m, $\delta_v = 0.1\frac{\mathrm{m}}{\mathrm{s}}$, $\delta_M = 100$ Nm.

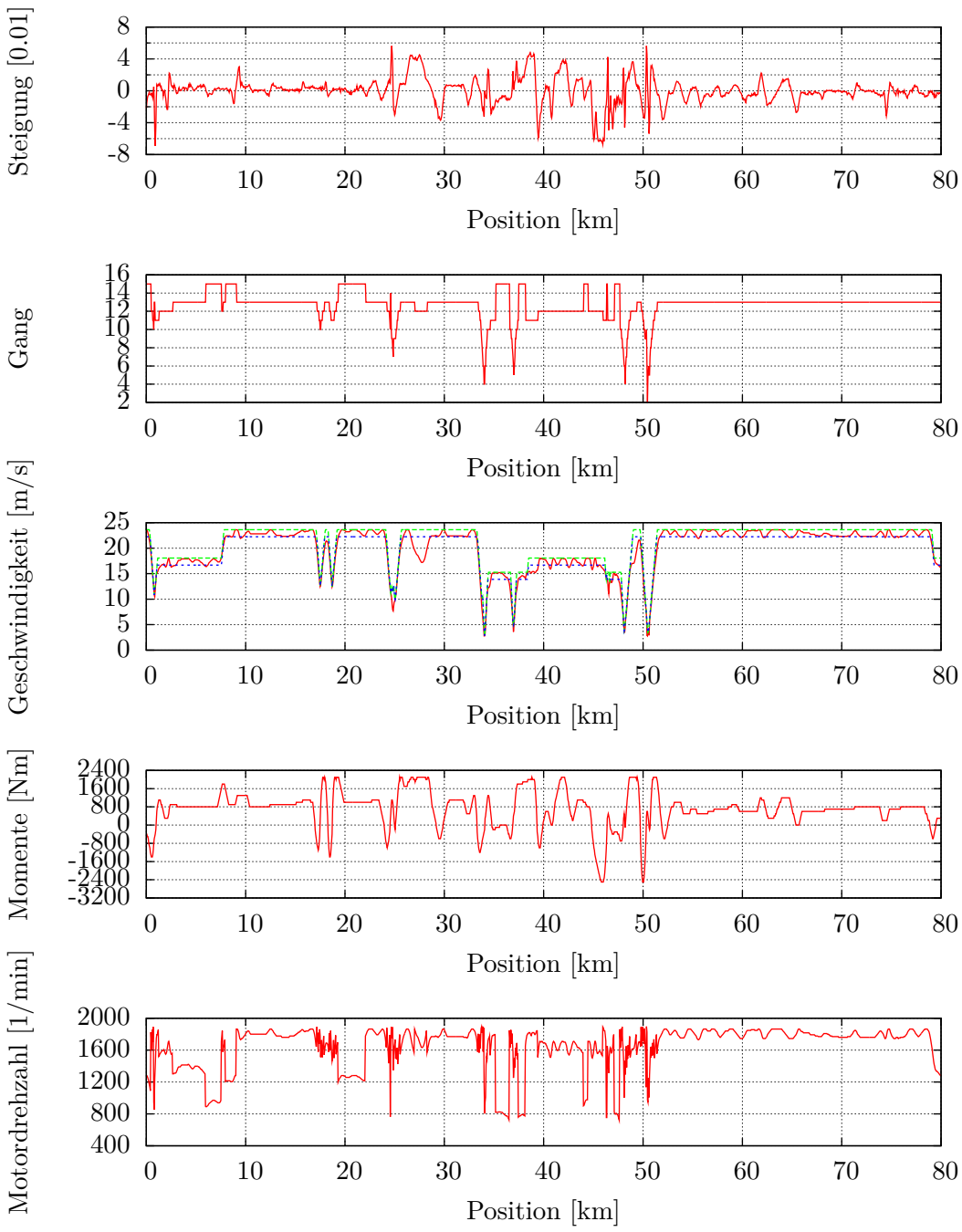

Abbildung 7.2: Ergebnisse auf einer Strecke von 80 km Länge mit Abschnitten reduzierter Höchstgeschwindigkeit. $\delta_s = 50$ m, $\delta_v = 0.1\frac{\mathrm{m}}{\mathrm{s}}$, $\delta_M = 100$ Nm.

7.1.2 Strecke mit variabler Maximalgeschwindigkeit

In diesem Abschnitt wird eine Strecke gewählt, in der Kurven auftreten, deren Krümmungen groß genug sind, um die Maximalgeschwindigkeit abschnittsweise – teilweise stark – herabzusetzen. Das Streckenprofil ist in Abbildung 7.2 zu sehen.

Zwei Phänomene können auf dieser Strecke sehr deutlich beobachtet werden: Zum einen wird auf Streckenabschnitten, auf denen Wunsch- und Maximalgeschwindigkeit konstant sind, aufgrund der Penalisierung des Schaltvorgangs nur sehr selten der Gang gewechselt. Um die Geschwindigkeit bei sich verändernder Steigung trotzdem zwischen Wunsch- und Höchstgeschwindigkeit zu halten, wird stattdessen die Steuerung des Motor- bzw. Bremsmoments benutzt. Zum anderen ist zu beobachten, dass auf diesen Abschnitten häufig nur im 14. Gang gefahren wird, anstatt noch weiter hochzuschalten. Möglicherweise ist das dem Umstand geschuldet, dass nicht konstant ohne mehrfaches Schalten bei sich ändernder Steigung im 15. oder 16. Gang gefahren werden könnte. Durch einen höheren Gang würde allerdings die Motordrehzahl gesenkt werden, was wiederum zu einem geringen Treibstoffverbrauch führen würde.

An dieser Stelle ist ein Verhalten zugunsten des Treibstoffverbrauchs durch eine andere Wahl der Zielfunktionsgewichte a_i zu erwarten. Diese Thematik wird in Abschnitt 7.4.4 angesprochen.

Im weiteren Verlauf dieser Arbeit werden wir exemplarisch einen Ausschnitt *dieser* Strecke für die Vergleichsrechnungen verwenden, da sie aufgrund der höheren Vielfalt geeigneter scheint, als die in Abschnitt 7.1.1 vorgestellte.

7.2 Vergleich zur Erweiterung Gleitender Horizont

An dieser Stelle wollen wir darauf eingehen, inwiefern sich die Ergebnisse der „vollständigen“ *Dynamischen Programmierung* von den Ergebnissen unterscheiden, die durch die Technik des *Gleitenden Horizonts*[20] berechnet wurden, vgl. Abschnitt 6.2. Mit „vollständig“ ist an dieser Stelle gemeint, dass das Problem global auf der kompletten Strecke optimiert wird.

Wir stellen Ergebnisse des *Gleitenden Horizonts* für verschiedene Längen T des Prädiktionshorizonts vor, um die Auswirkung dieser Größe besser verstehen zu können.

Für die Bewertung der Ergebnisse von Algorithmus 4 gehen wir wie folgt vor: Die Kosten der angewandten Steuerungen – d.h. jeweils der Steuerung, die den Anfang einer für die Länge des Prädiktionshorizonts optimalen Steuerung darstellt – werden aufsummiert. Hinzu kommen die Kosten für den gesamten letzten Prädiktionshorizont, das Endstück der Länge T.

Allen Berechnungen in diesem Abschnitt haben wir die gleiche Strecke zugrunde gelegt. Es ist ein Ausschnitt aus der bereits in Abschnitt 7.1.2 betrachteten Strecke. Diese scheint

[20] Siehe Kapitel 4.

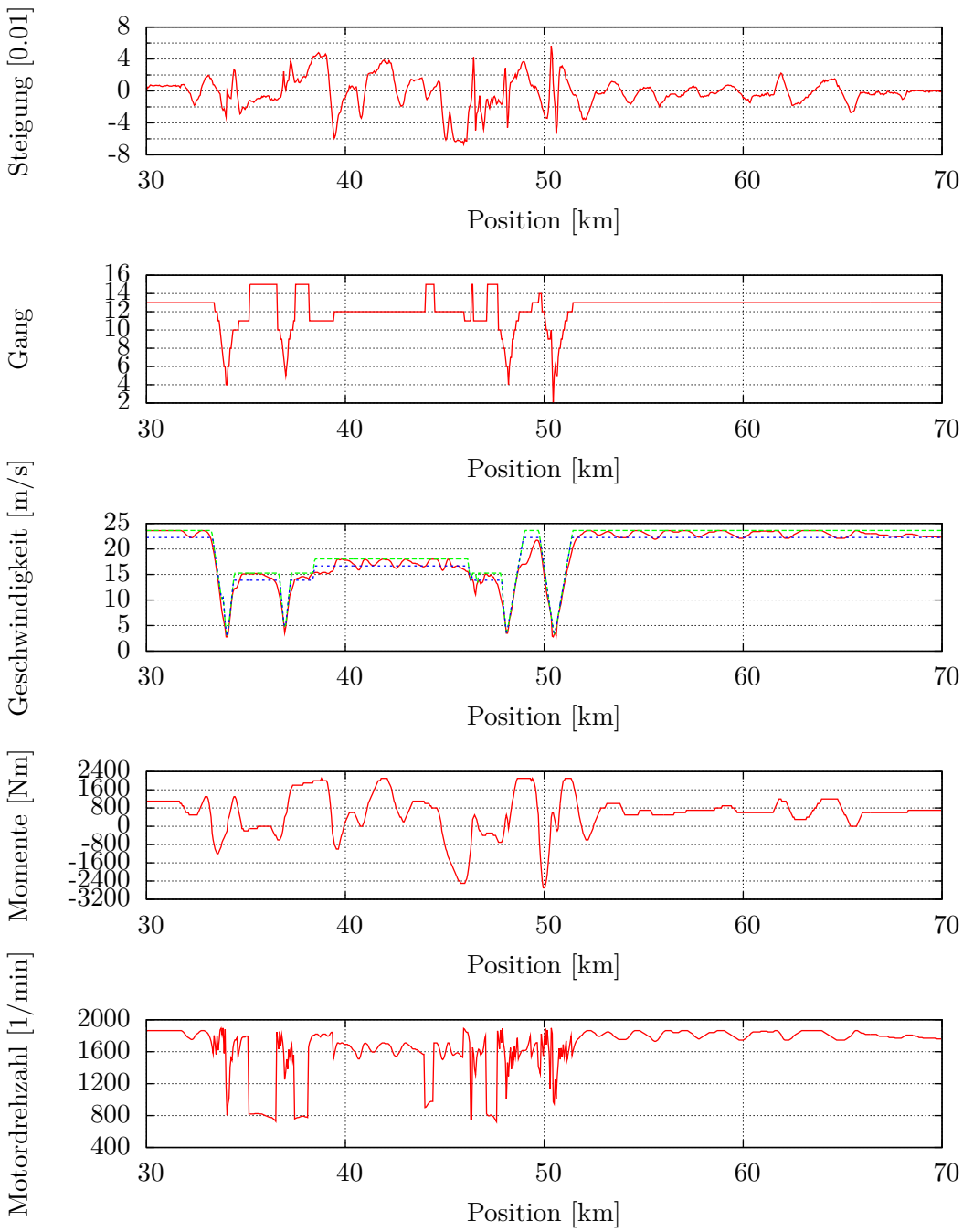

Abbildung 7.3: Globales Optimum auf einer 40 km langen Vergleichsstrecke. $m = 40$ t $\delta_s = 50$ m, $\delta_v = 0.1\frac{\mathrm{m}}{\mathrm{s}}$, $\delta_M = 100$ Nm.

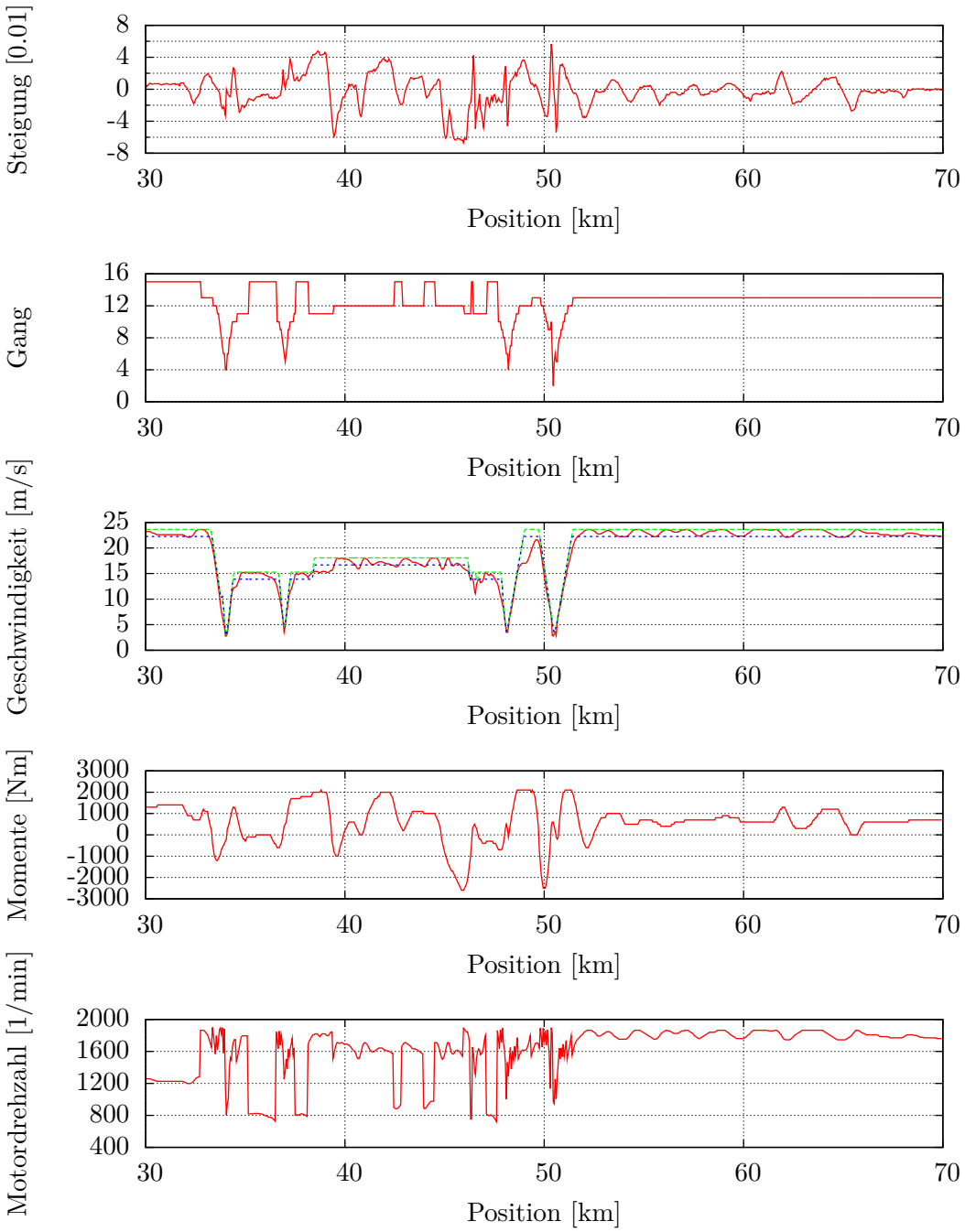

Abbildung 7.4: Die gleiche Strecke wie in Abbildung 7.3, hier mit einem Prädiktionshorizont von $T = 1000$ m. $\delta_s = 50$ m, $\delta_v = 0.1\frac{\mathrm{m}}{\mathrm{s}}$, $\delta_M = 100$ Nm.

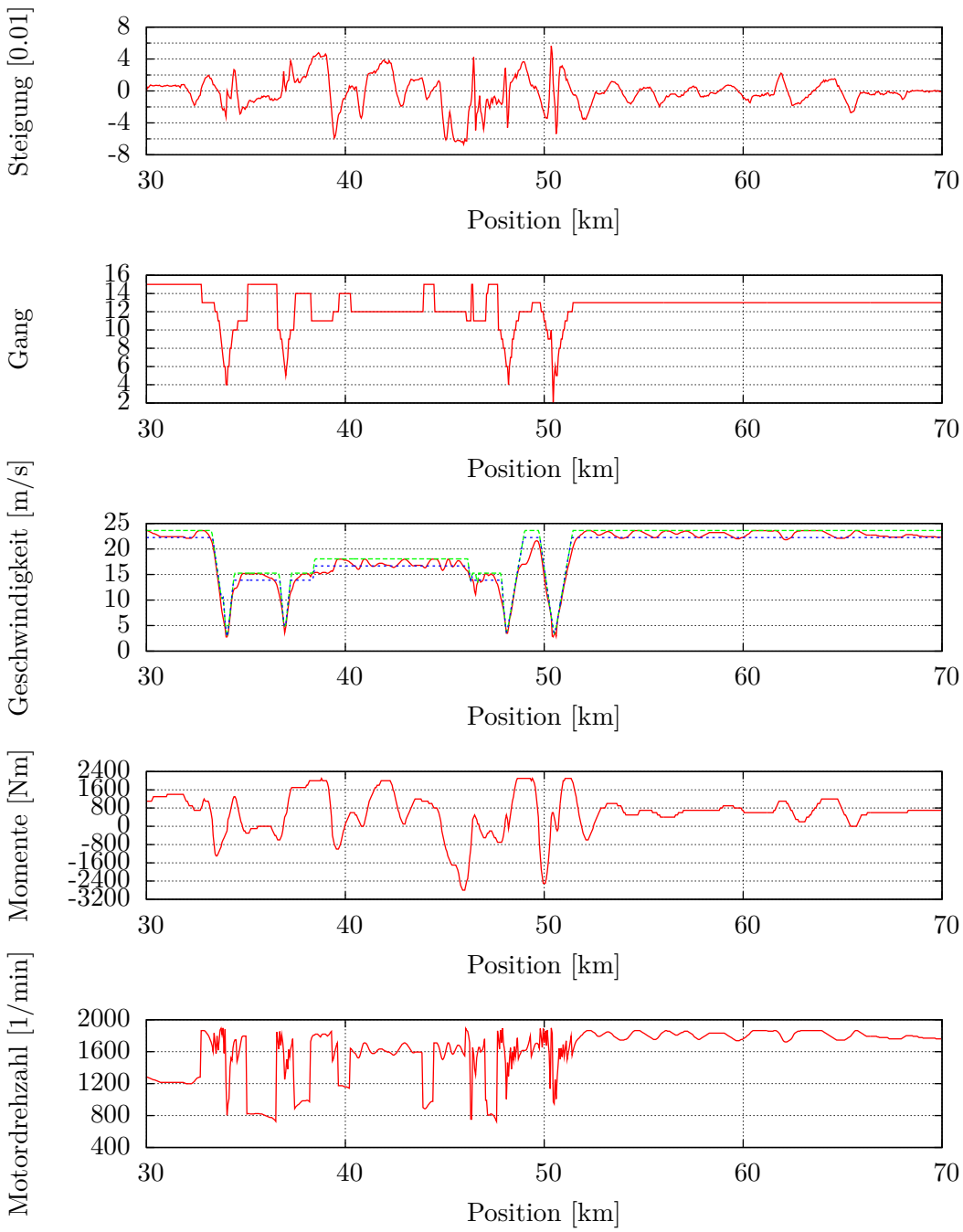

Abbildung 7.5: Die gleiche Strecke wie in Abbildung 7.3, hier mit einem Prädiktionshorizont von $T = 800$ m. $\delta_s = 50$ m, $\delta_v = 0.1\frac{\mathrm{m}}{\mathrm{s}}$, $\delta_M = 100$ Nm.

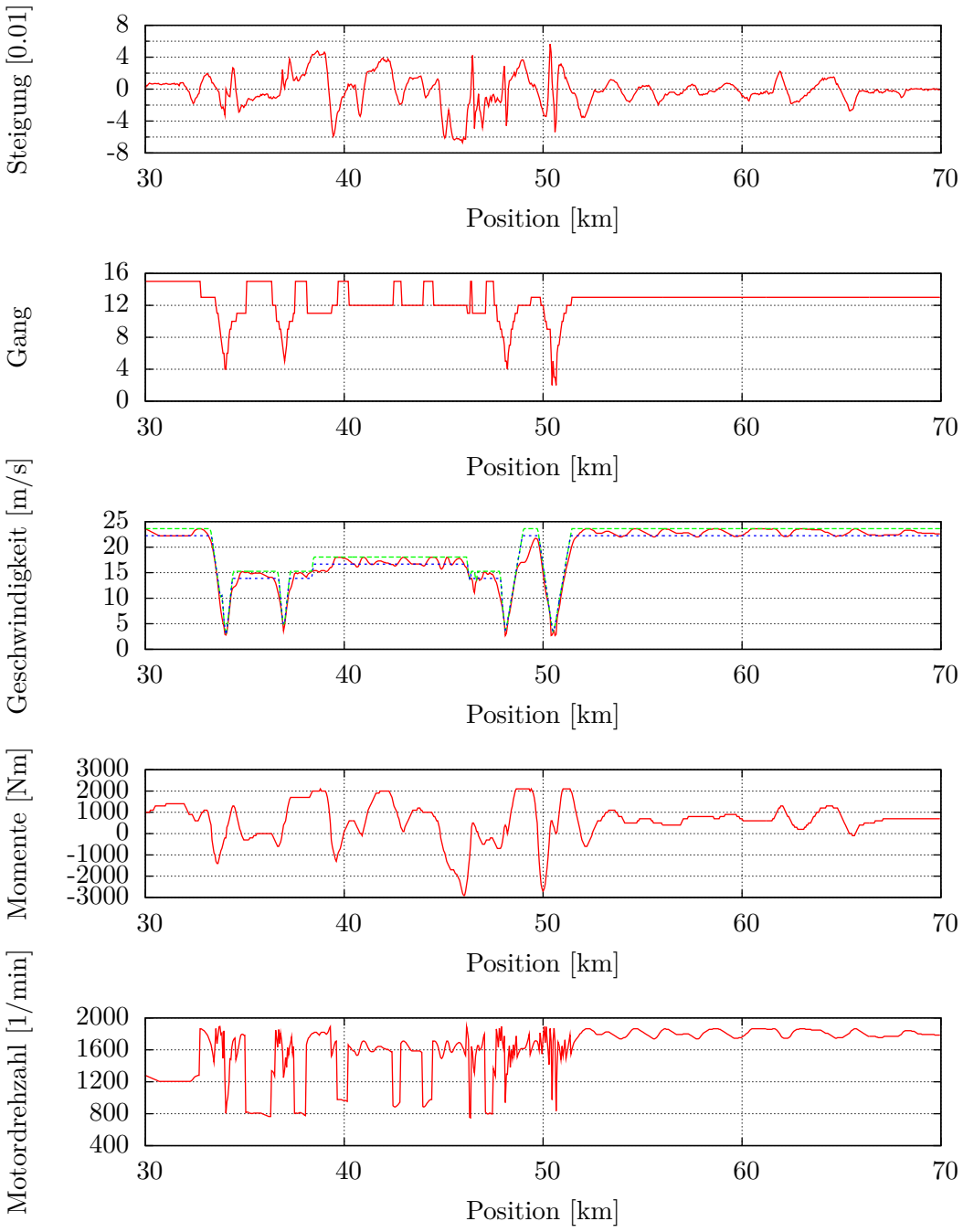

Abbildung 7.6: Die gleiche Strecke wie in Abbildung 7.3, hier mit einem Prädiktionshorizont von $T = 600$ m. $\delta_s = 50$ m, $\delta_v = 0.1\frac{\mathrm{m}}{\mathrm{s}}$, $\delta_M = 100$ Nm.

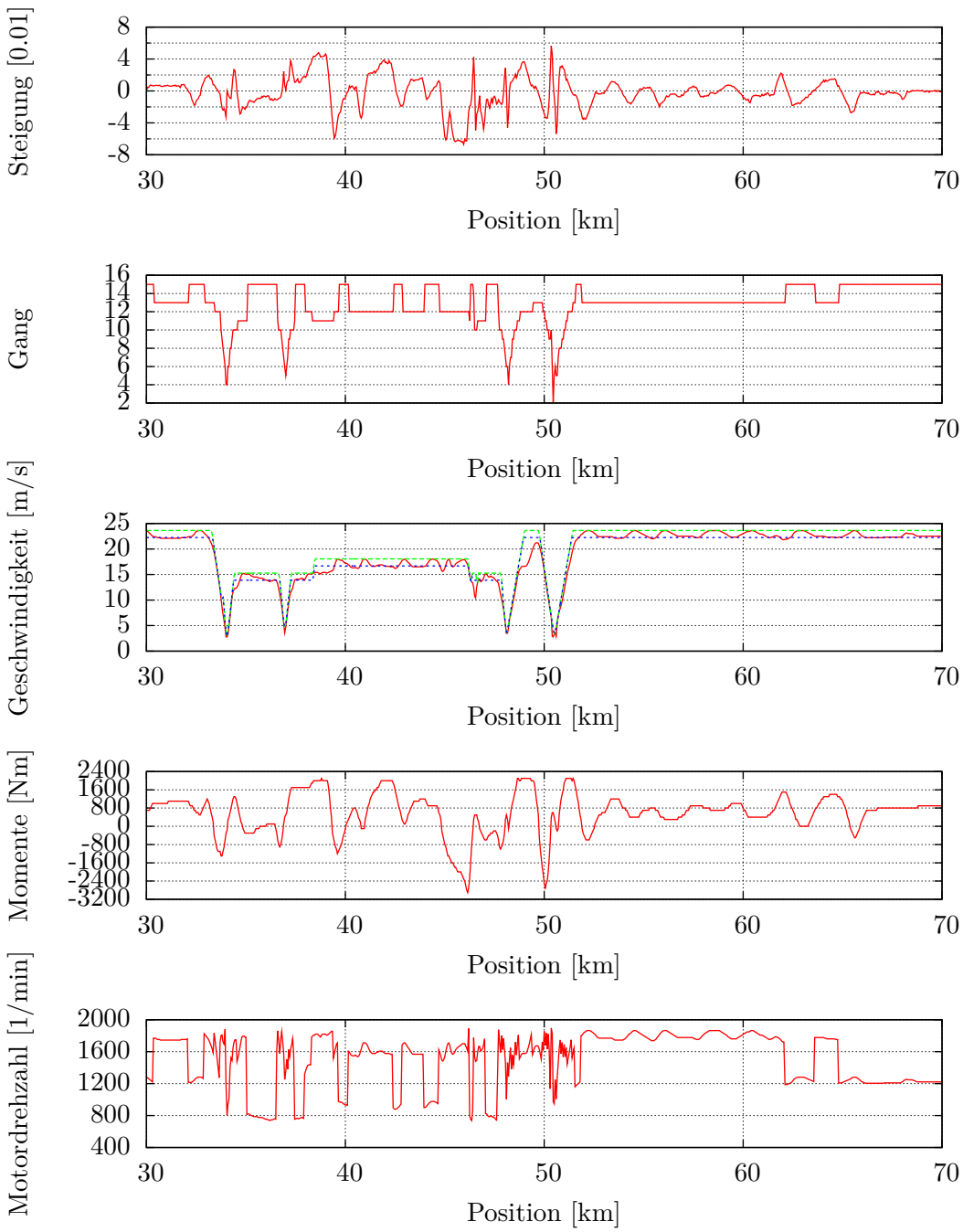

Abbildung 7.7: Die gleiche Strecke wie in Abbildung 7.3, hier mit einem Prädiktionshorizont von $T = 400$ m. $\delta_s = 50$ m, $\delta_v = 0.1\frac{\mathrm{m}}{\mathrm{s}}$, $\delta_M = 100$ Nm.

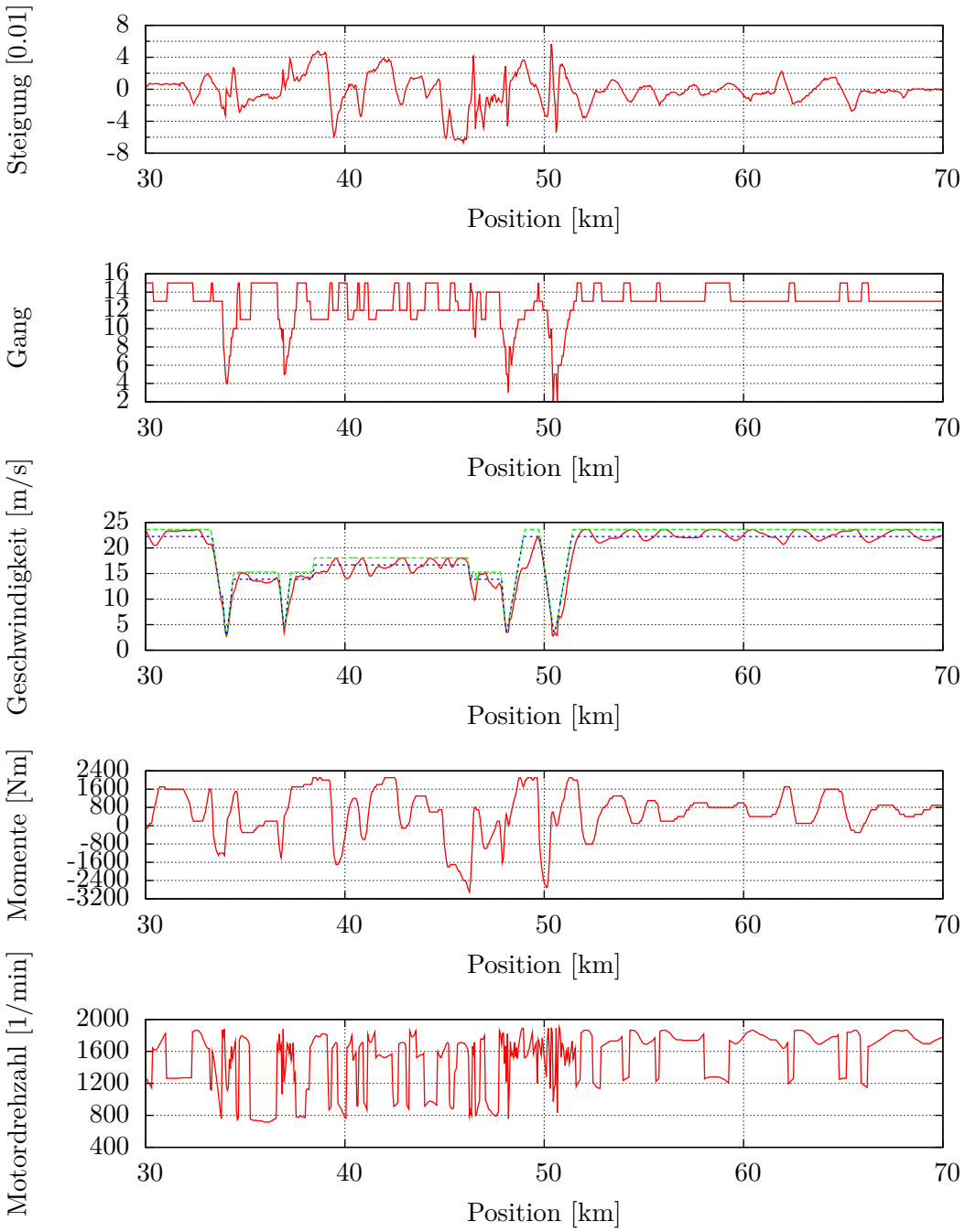

Abbildung 7.8: Die gleiche Strecke wie in Abbildung 7.3, hier mit einem Prädiktionshorizont von $T = 200$ m. $\delta_s = 50$ m, $\delta_v = 0.1\frac{\mathrm{m}}{\mathrm{s}}$, $\delta_M = 100$ Nm.

für die Analyse geeignet, da man an den Stellen, an denen die Maximalgeschwindigkeit reduziert ist, beobachten kann, wie *vorausschauend* eine Steuerung ist.

In Abbildung 7.3 ist das globale Optimum auf dieser Strecke zu sehen. Im Vergleich dazu sieht man in Abbildung 7.4 die Optimalsteuerung für einen gleitenden Prädiktionshorizont von 1000 m Länge.

Überraschenderweise sieht man für $T = 1000$ m nur minimale Unterschiede zur global optimalen Steuerung über die 40 km lange Vergleichsstrecke. Dementsprechend weicht der Zielfunktionswert auch nur um weniger als 0.1 % vom Optimum ab.

Auch bei einer Verkürzung des Prädiktionshorizonts auf $T = 800$ m ist das Ergebnis von Algorithmus 4 im Vergleich zur global optimalen Lösung befriedigend. Der Zielfunktionswert wächst um weniger als 1 % gegenüber dem globalen Optimum und es sind keine signifikanten Unterschiede in der Steuerung zu den vorher diskutierten Optimalsteuerungen zu sehen.

Wenn man die Optimallösung auf der gleichen Strecke mit einem Prädiktionshorizont von lediglich 400 Metern Länge berechnet, ergeben sich dagegen schon sichtbare Veränderungen, siehe Abbildung 7.7. So treten wesentlich öfter Schaltvorgänge auf, die zu vermeiden gewesen wären, wenn ein größeres Stück der Strecke bekannt gewesen wäre. Als Beispiel kann man den relativ „gemäßigten“ Abschnitt ab Kilometer 52 heranziehen. Schon in der Rechnung mit $T = 1000$ ist hier kein Schaltvorgang mehr erforderlich, da rechtzeitig vor den leichten Steigungen auf die zulässige Höchstgeschwindigkeit beschleunigt wird. Bei $T = 400$ wird kurzfristig in den höchsten modellierten Gang geschaltet, da die Strecke der nächsten 400 Meter vollständig aus Gefälle besteht und keine Notwendigkeit erkannt wird, später wieder herunterschalten zu müssen.

Noch deutlicher erkennbar sind die gerade beschriebenen Effekte bei einer weiteren Reduzierung von T auf 200 m. Wenn man den Plot der gewählten Gänge betrachtet, sieht man nun eine weiter gesteigerte Anzahl an Schaltvorgängen, die durch längere Prädiktionshorizonte vermieden werden konnte. Mit 200 m Länge ist der Prädiktionshorizont so kurz, dass auf dieser Strecke auf dem ruhigeren Abschnitt ab Streckenkilometer 52 trotz häufigeren Schaltens die Wunschgeschwindigkeit mehrfach unterschritten wird. Allein die Kombination dieser beiden Phänomene führt schon zu einer erheblichen Erhöhung des Zielfunktionswerts. Man kann auch sehen, dass die Kurve der Momente höhere Ausschläge in beiden Richtungen zeigt, was für $T = 400$ m noch nicht so stark zu beobachten war. Auch dieses Verhalten ist unerwünscht und wird in der Zielfunktion über die Änderung der Momente bestraft. All diese Effekte schlagen sich am Ende dann auch im Zielfunktionswert nieder. Dieser liegt mit 3560.873419 fast doppelt so hoch wie das maximal zu erreichende globale Optimum. Das Ergebnis ist daher völlig unbrauchbar.

Eine Besonderheit, die bei den Prädiktionslängen 200 m und 400 m auftritt, ist das „Verpassen“ des Wiederanstiegs der Wunschgeschwindigkeit nach starkem Abfall kurz nach Streckenkilometer 50. Bei den genannten Steuerungen bleibt die tatsächliche Geschwindigkeit konstant unterhalb der Wunschgeschwindigkeit, weil der Prädiktionshorizont zu kurz ist, um erkennen zu können, dass der Anstieg noch weiter anhält.

Länge des Prädiktionshorizonts [m]	Zielfunktionswert	Abbildung
40000 (global)	1892.324674	7.3
1000	1893.793579	7.4
800	1905.680392	7.5
600	1996.831971	7.6
400	2126.637915	7.7
200	3560.873419	7.8

Tabelle 7.1: Übersicht der Zielfunktionswerte für verschiedene Längen T des Prädiktionshorizonts im Vergleich zum globalen Optimum.

Abschließend bleibt festzuhalten, dass – wie zu erwarten war – die Ergebnisse auf die ganze Strecke gesehen mit abnehmender Länge des Prädiktionshorizonts immer schlechter werden. Allerdings ist doch interessant zu sehen, dass ein Horizont von 1000 m Länge bei der von uns gewählten Strecke ausreicht, um dem globalen Optimum auf 0.1 % nahe zu kommen. Erstaunlich ist weiterhin, dass auch für 800 m und 600 m Steuerungen berechnet wurden, deren Zielfunktionswert in vertretbarer Entfernung zum Optimum liegt, die Qualtät der Lösung mit weiterer Reduzierung aber dann rasant abnimmt.

In Tabelle 7.1 ist eine Übersicht über die Güte der jeweiligen Minima zu finden.

7.3 Vergleich zwischen Dynamischer Programmierung und einer direkten Methode

In diesem Abschnitt wollen wir die Ergebnisse der unterschiedlichen in Kapitel 2 angesprochenen Algorithmen vergleichen. Wir haben uns für eine gemeinsame Strecke entschieden, die mehrere Stellen aufweist, an denen aufgrund der Krümmung der Strecke sehr langsam gefahren werden muss, um auch das Beschleunigungsverhalten zu untersuchen.

Beide Algorithmen verwenden für die Berechnung der Optimalsteuerung einen *Gleitenden Horizont* von 1000 m Länge. Dieser Horizont wird für das *Direkte Mehrzielverfahren* in 20 Abschnitte à 50 m unterteilt. Ist eine optimale Lösung für den Horizont gefunden, wird dieser um 10 m verschoben, um neu zu iterieren.

Auch bei der *Dynamischen Programmierung* wird der Prädiktionshorizont in 20 Abschnitte zu jeweils 50 m unterteilt, allerdings wird hier der Horizont um die Diskretisierungslänge $\delta_s = 50$ m verschoben, wenn das Optimum auf dem Horizont gefunden worden ist, damit möglichst viele Daten aus der vergangenen Iteration auf das neue Gitter übertragen werden können.

Die Zielfunktion bestraft in diesem Vergleichsszenario einerseits den Treibstoffverbrauch und andererseits die quadratische Abweichung von der Wunschgeschwindigkeit –

im Gegensatz zur bisher formulierten Zielfunktion allerdings sowohl zu niedrige als auch zu hohe Geschwindigkeiten.

Die direkte Methode benutzt die in Abschnitt 3.3 vorgestellte *Konvexifizierung*, um die im Gang relaxierten Probleme zu lösen. Das in Abbildung 7.9 visualisierte Ergebnis stellt die Lösung dar, in der die Gänge noch ungerundet vorliegen. Um eine ganzzahlige Lösung zu erhalten, müsste noch eine geeignete Rundungsstrategie[21] auf das Ergebnis angewandt werden. Allerdings ist zu erkennen, dass durch die Konvexifizierung an vielen Stellen der gewählte Gang bereits ganzzahlig ist.

Betrachtet man den Plot der Gänge genauer, sieht man, dass an einigen Stellen sehr oft abwechselnd zwischen zwei Gängen geschaltet wird. Dies wird dadurch begünstigt, dass wir uns dazu entschieden haben, in diesem Vergleich die Anzahl der Schaltvorgänge nicht zu limitieren oder zu bestrafen. Bei Verwendung einer direkten Methode bestände für die einfache Bestrafung von Schaltvorgängen das Problem, dass diese Zielfunktion nicht differenzierbar ist. Das Problem besteht für die *Dynamische Programmierung* zwar nicht, aber damit die Ergebnisse vergleichbar bleiben, haben wir von einem Strafterm bei der *Dynamischen Programmierung* ebenfalls abgesehen.

Abbildung 7.10 zeigt das Ergebnis der *Dynamischen Programmierung*. Analysiert man den Verlauf der gewählten Gänge, fällt auf, dass hier seltener geschaltet wird, obwohl wir – wie soeben erwähnt – dieses Verhalten nicht explizit über die Zielfunktion oder die Nebenbedingungen gefordert haben. Dies kann einerseits an der höheren Auflösung der Daten der direkten Methode liegen,[22] hat sicherlich aber auch algorithmische Gründe, da die Lösung in Abbildung 7.10 zwischen Kilometer 26 und 30 nur noch ein mal kurz in einen tieferen Gang wechselt, während beim Ergebnis der direkten Methode auch auf diesem Streckenabschnitt sehr häufig der Gang gewechselt wird.

Insgesamt fällt auf, dass beim Ergebnis der direkten Methode die Kurve des Gangs hauptsächlich aus zwei Mustern besteht. Zum einen sind das Streckenabschnitte, auf denen der Gang konstant gehalten wird – wie etwa ab Kilometer 35 – und zum anderen gibt es Abschnitte, bei denen der Gang sehr schnell und wild gewechselt wird. Nur sehr vereinzelt treten Phasen auf, in denen klar strukturierte Gangwechsel erfolgen. Die Abschnitte der häufigen Gangwechsel sind in der Regel diejenigen, auf denen sich die Fahrzeugdynamik – aufgrund von teilweise starken Änderungen der Maximalgeschwindigkeit – ändern muss. Dies sind auch die Stellen, an denen vermehrt nichtganzzahlige Gänge als optimal angesehen werden.

Eine Gemeinsamkeit beider Algorithmen in Bezug auf die Gangwahl stellt die nur sehr seltene Nutzung des Gangs 15 dar.[23] Von beiden Algorithmen werden die Gänge 14 und 16 häufig gewählt und es wird auch oft zwischen diesen gewechselt, allerdings fast immer ohne durch Gang 15 zu schalten. Diese bemerkenswerte Tatsache ist wohl auf die

[21] Siehe Abschnitt 3.4.2.
[22] Bei der direkten Methode liegen die Daten alle 10 m anstatt nur alle 50 m vor.
[23] In den Plots ist dieser Gang mit 14 nummeriert.

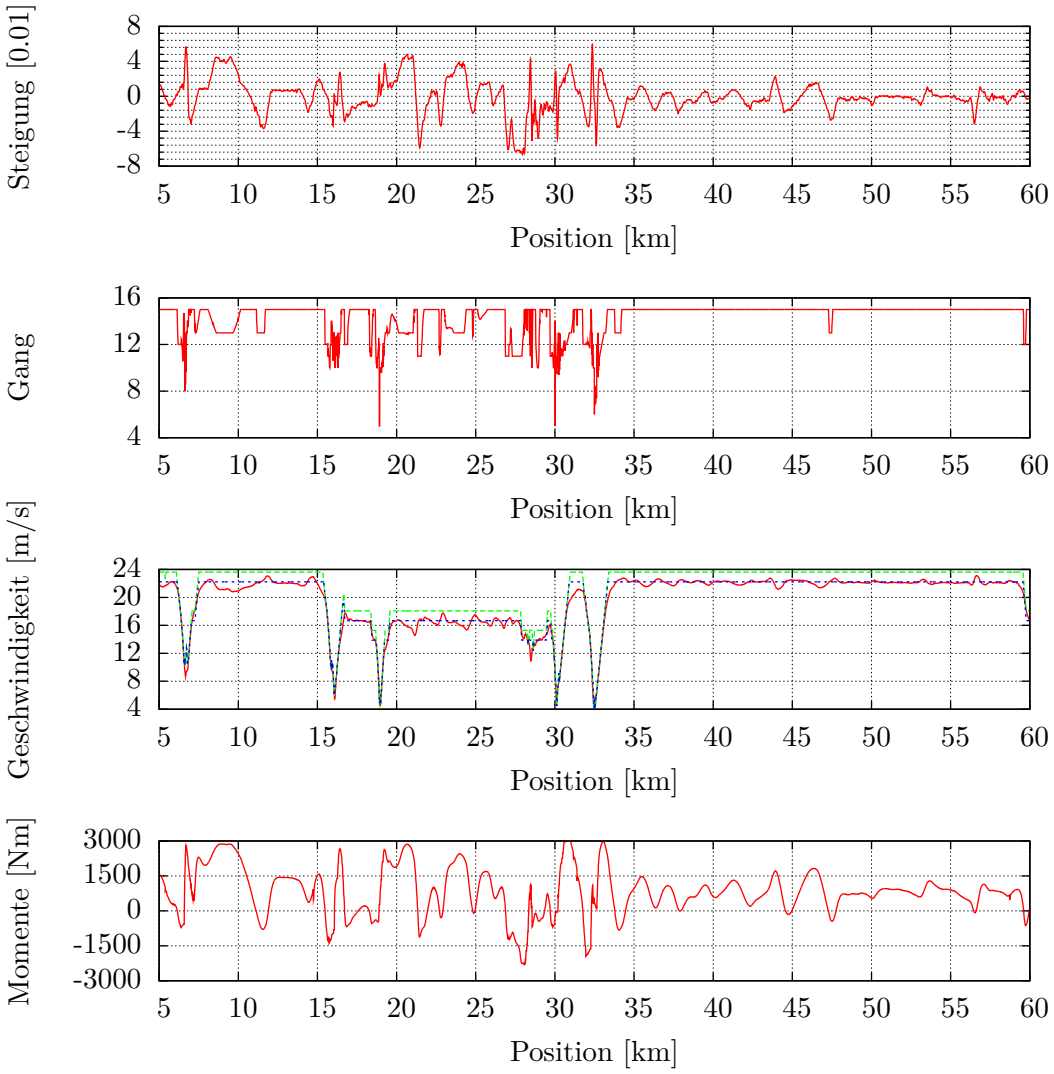

Abbildung 7.9: Optimalsteuerung einer direkten Methode mit 1000 m langem gleitendem Horizont.

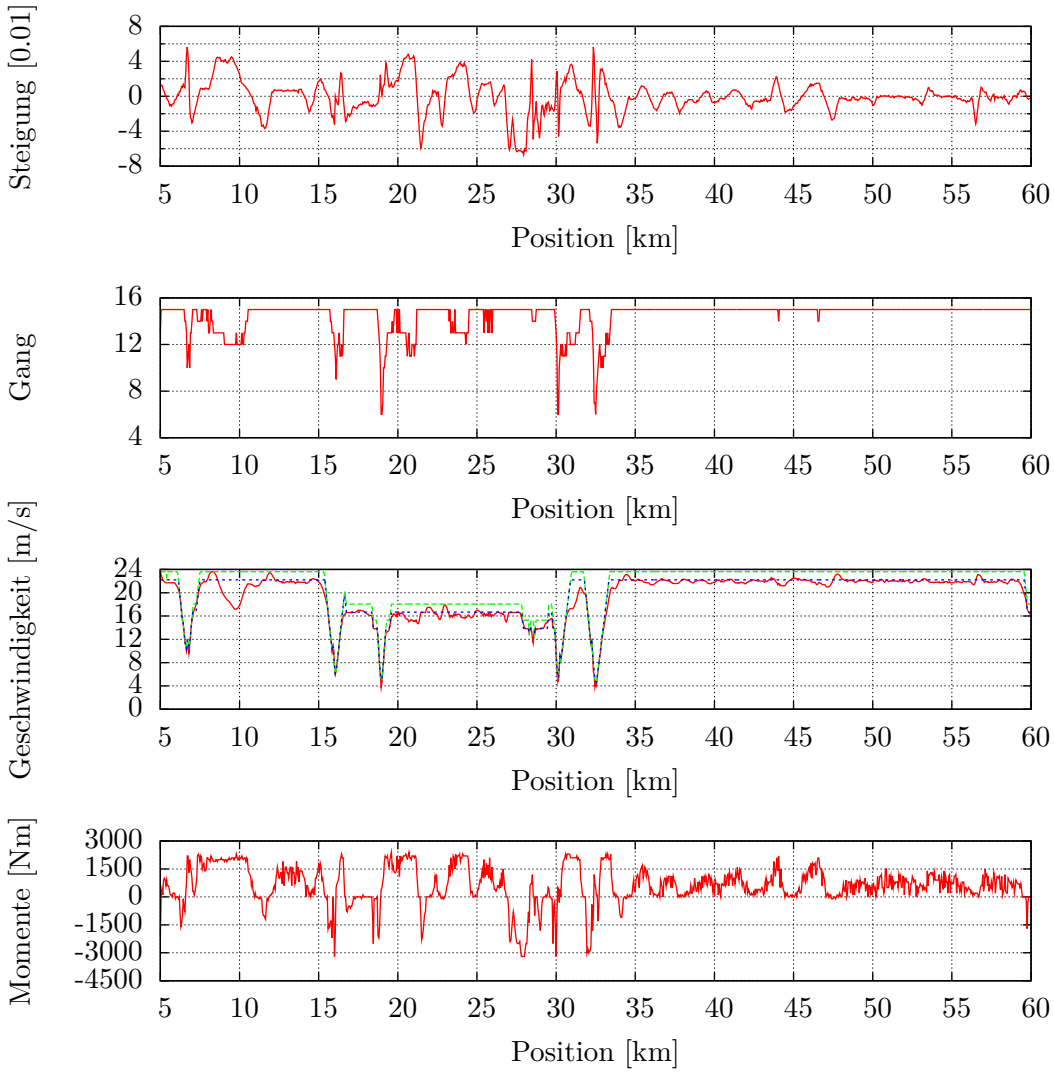

Abbildung 7.10: Mit Algorithmus 4 berechnetes Optimum, ebenfalls mit 1000 m langem gleitendem Horizont. $\delta_s = 50$ m, $\delta_v = 0.1\frac{\mathrm{m}}{\mathrm{s}}$, $\delta_M = 100$ Nm.

Verhältnisse der Konstanten $i_T(y)$ (Übersetzungsverhältnis) und $\eta_T(y)$ (Wirkungsgrad) im gemeinsamen Modell zurückzuführen.[24] Dabei stellt sich die Frage, ob das Modell lediglich ungenau ist, oder ob es wirklich selten sinnvoll ist, diesen Gang zu wählen.

Die Geschwindigkeitstrajektorien beider Ergebnisse ähneln sich aufgrund der in der Zielfunktion verankerten Bestrafung der Abweichung von der Wunschgeschwindigkeit sehr und verlaufen in den meisten Streckenabschnitten sehr nah an der Wunschgeschwindigkeit. Allerdings zeigen beide Ergebnisse auch ein leichtes Schwanken sowohl ober- als auch unterhalb der Wunschgeschwindigkeit auf.

Erwähnenswert ist allerdings die Stelle kurz vor Kilometer 7, an der sowohl eine starke Steigung von kurzzeitig bis zu 6 % vorliegt, als auch die Maximalgeschwindigkeit zeitweise auf 11 $\frac{\mathsf{m}}{\mathsf{s}}$ herabgesetzt wird. Auf diesem speziellen Streckenabschnitt gelingt es der direkten Methode nicht ganz so gut wie der *Dynamischen Programmierung*, so zu steuern, dass Maximal- und Wunschgeschwindigkeit beachtet werden. Stattdessen fällt die Geschwindigkeit etwas stärker als bei der *Dynamischen Programmierung* bis auf unter 9 $\frac{\mathsf{m}}{\mathsf{s}}$ ab, was nicht sinnvoll erscheint, da der LKW danach wieder auf normale Geschwindigkeiten überhalb von 20 $\frac{\mathsf{m}}{\mathsf{s}}$ beschleunigt werden muss.

Aufgrund der niedrigeren Geschwindigkeit im Gegensatz zum Ergebnis der *Dynamischen Programmierung* wird an dieser Stelle zwangsläufig auch bis in Gang 9 heruntergeschaltet, während Gang 11 laut *Dynamischer Programmierung* ausreichen würde, um den LKW diese Passage durchfahren zu lassen.

Bei den vier ähnlichen Abschnitten dieser Strecke, in denen die Geschwindigkeit aufgrund der niedrigen Maximalgeschwindigkeit ebenfalls reduziert werden muss, generieren beide Algorithmen dagegen ähnliche Ergebnisse.

Eine nicht ganz selbstverständliche Gemeinsamkeit, die beide Ergebnisse innehaben und die oben schon kurz angesprochen wurde, ist die Tatsache, dass die Geschwindigkeitstrajektorien zeitweise oberhalb der Wunschgeschwindigkeit liegen. Aufgrund der hier verwendeten Wahl der Zielfunktion, die dieses Verhalten durch die quadratische Abweichung direkt und durch den Treibstoffverbrauch indirekt bestraft, ist dies auf den ersten Blick erstaunlich. Untersucht man dieses Verhalten aber genauer, kann man feststellen, dass dies häufig vor oder zu Beginn von steilen Anstiegen auftritt, wie es etwa für die Ergebnisse beider Algorithmen bei Kilometer 8 und 23 zu erkennen ist. Hier ist es – bezüglich der hier verwendeten Zielfunktion – günstiger, die Geschwindigkeit kurzfristig zu überschreiten und danach leicht zu unterschreiten, als die Geschwindigkeit so lange wie möglich zu halten, um dann eine starke Abweichung nach unten in Kauf nehmen zu müssen.

Vergleicht man die Kurven der Momente, findet man eine Erklärung für den – trotz der geringeren örtlichen Auflösung – glatten Verlauf der Geschwindigkeit beim Ergebnis der *Dynamischen Programmierung*. Da Änderungen der Momente nicht bestraft werden, kann hier beliebig gesteuert werden, was im Fall der *Dynamischen Programmierung* auch

[24] Welchen Einfluss diese Größen in dem von uns gewählten Modell haben, ist in Kapitel 5 nachzulesen.

stärker ausgenutzt wird. Die Momente folgen einem sehr zackigen Verlauf, der fast zu jedem neuen Diskretisierungspunkt einen neuen Wert annimmt. Durch diese sehr feine Steuerung gelingt es, die Geschwindigkeit nah an der Wunschgeschwindigkeit zu halten. Durch die sehr schnell reagierende Steuerung der Momente wird hier ebenfalls ermöglicht, dass auf dem Streckenabschnitt ab Kilometer 35 nicht mehr gebremst werden muss.

Es bleibt festzuhalten, dass die hier verwendete direkte Methode – die im Gegensatz zur *Dynamischen Programmierung* echtzeitfähig ist – auf weiten Teilen der Strecke vergleichbare Ergebnisse liefert, es aber einzelne Abschnitte gibt, in denen sich – so scheint es – das erreichte lokale Optimum auf dem Prädiktionshorizont leicht vom globalen Optimum unterscheidet. An diesen Stellen liefert die *Dynamische Programmierung* eine qualitativ bessere Lösung.

Beim Vergleich der beiden Ergebnisse sollte weiterhin nicht vergessen werden, dass das Ergebnis der direkten Methode hier noch nicht gerundet ist und daher die Möglichkeit nutzt – den ganzzahligen Gängen an diesen Stellen offensichtlich überlegene – nichtganzzahlige Gänge einzulegen.

Das Ergebnis der *Dynamischen Programmierung* lag nach 40 Minuten vor, während die direkte Methode lediglich 50 Sekunden brauchte, um ihr Ergebnis zu liefern. Beide Rechnungen fanden auf vergleichbaren Rechnerarchitekturen statt.

7.4 Sensitivitätsanalyse

Wie sehr das Ergebnis der *Dynamischen Programmierung* von verschiedenen Eingangsgrößen abhängt, werden wir in diesem Abschnitt untersuchen. Zu den zu analysierenden Größen gehören einerseits Modellparameter wie die Masse des LKW und Gegenwind, andererseits aber auch algorithmische Einstellungsparameter wie die Gewichtung der einzelnen Kriterien in der Zielfunktion und die Feinheit der Diskretisierungen der Strecke, Zustände und Steuerungen.

7.4.1 Masse

Anhand von Berechnungen optimaler Steuerungen auf einer Vergleichsstrecke, bei denen jeweils nur die Masse des Fahrzeugs im Modell verschieden gewählt wurde, wollen wir den Einfluss dieser Größe auf die Fahrweise untersuchen. Wir haben die Berechnung mit $m \in \{30, 35, 40, 42, 45\}$ t durchgeführt. Zu sehen sind die Ergebnisse in den Abbildungen 7.11 bis 7.14. Als Referenzlösung, die mit 40 t Gesamtgewicht gerechnet wurde, dient Abbildung 7.3.

Als Ergebnis der Untersuchung der Massenabhängigkeit ist festzuhalten, dass die Geschwindigkeitstrajektorien sich nicht signifikant von einander unterscheiden. Erst bei großen Masseunterschieden – 30 t zu 45 t – sind kleine Nichtübereinstimmungen zu sehen. So erreicht bei einer Masse von 45 t ungefähr bei Streckenkilometer 38 die Geschwindigkeit nicht direkt die Wunschgeschwindigkeit, so wie es beim leichteren LKW

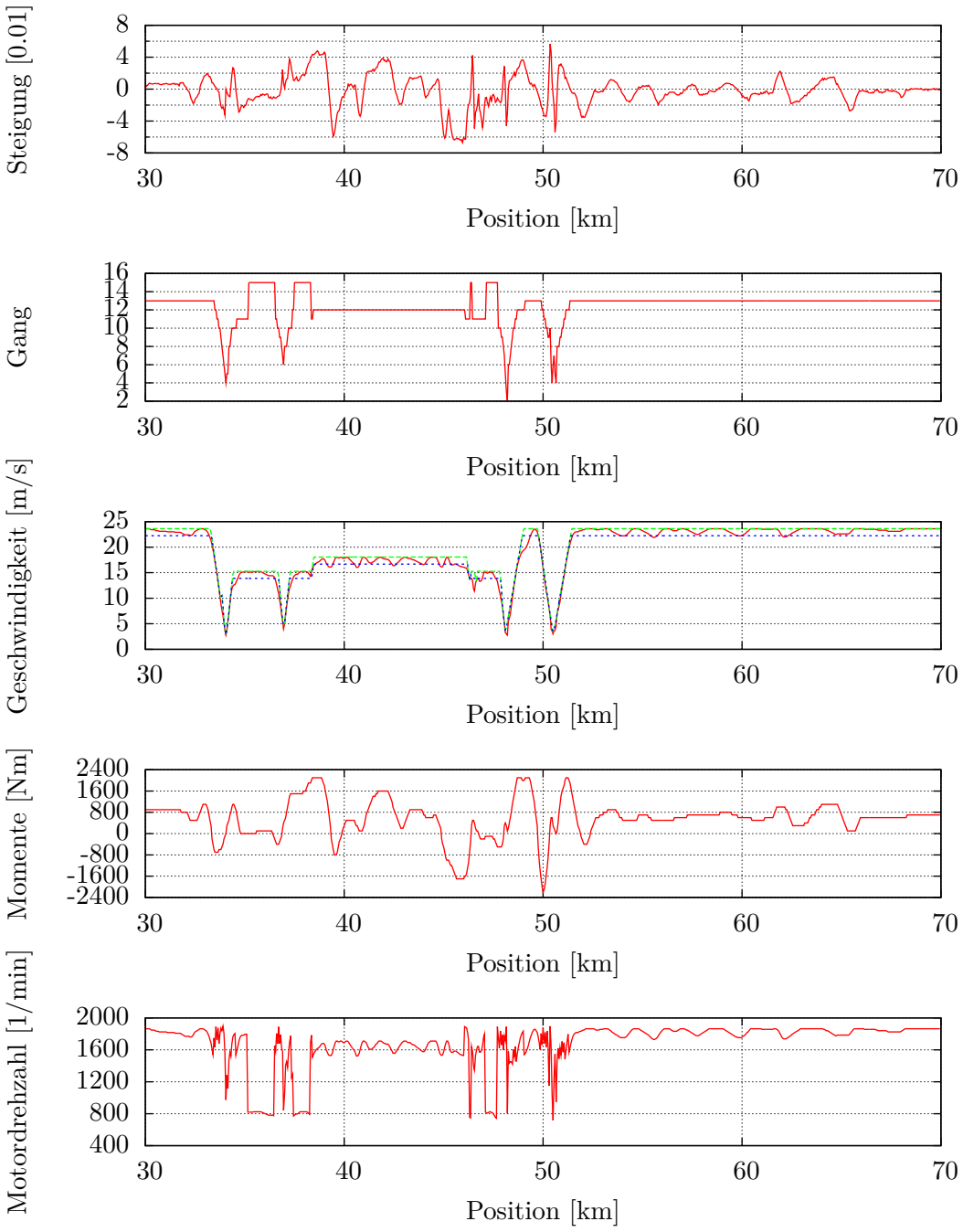

Abbildung 7.11: Optimalsteuerung bei einer Fahrzeugmasse von 30 t. $\delta_s = 50$ m, $\delta_v = 0.1\frac{\mathrm{m}}{\mathrm{s}}$, $\delta_M = 100$ Nm.

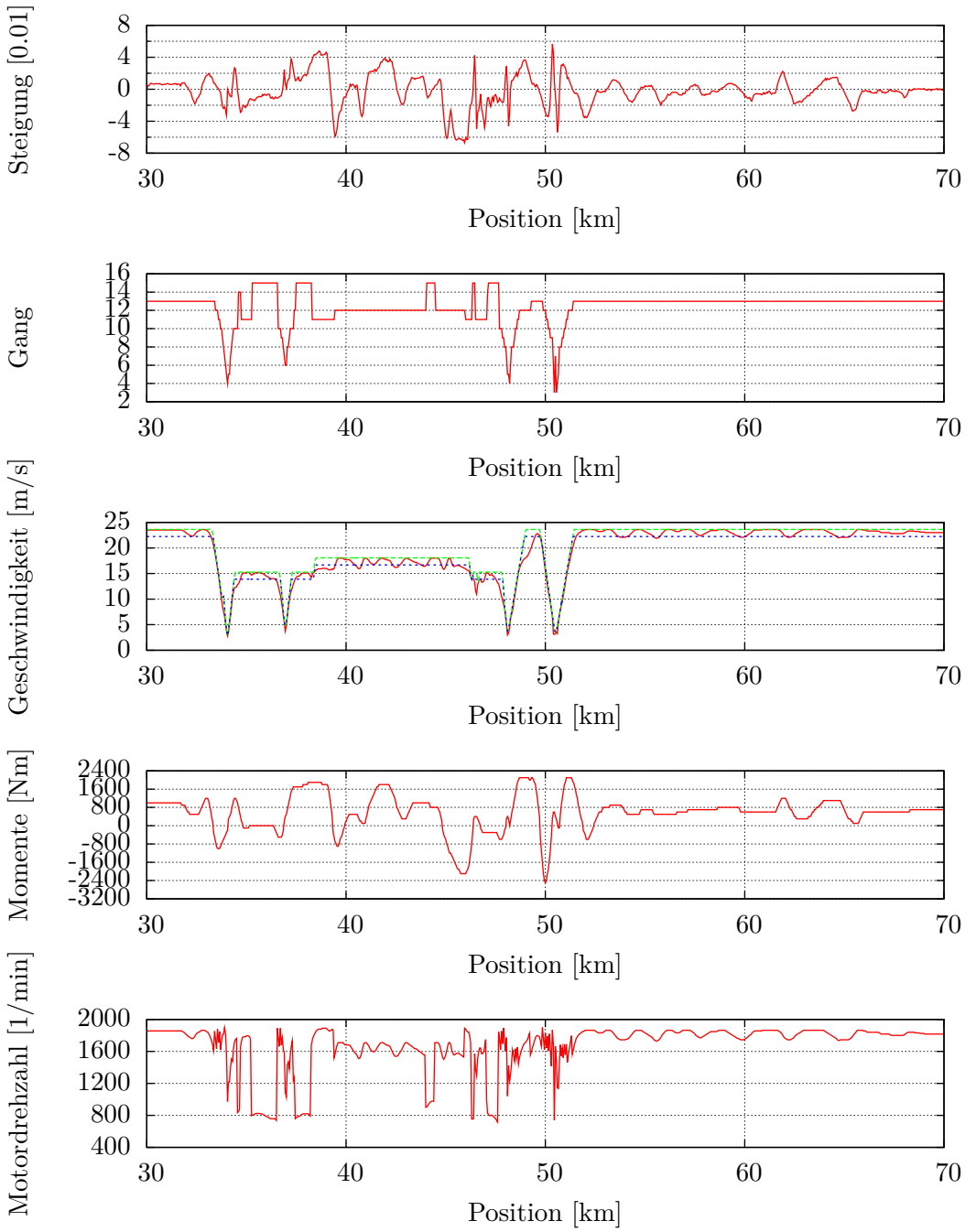

Abbildung 7.12: Optimalsteuerung bei einer Fahrzeugmasse von 35 t. $\delta_s = 50$ m, $\delta_v = 0.1\frac{\mathrm{m}}{\mathrm{s}}$, $\delta_M = 100$ Nm.

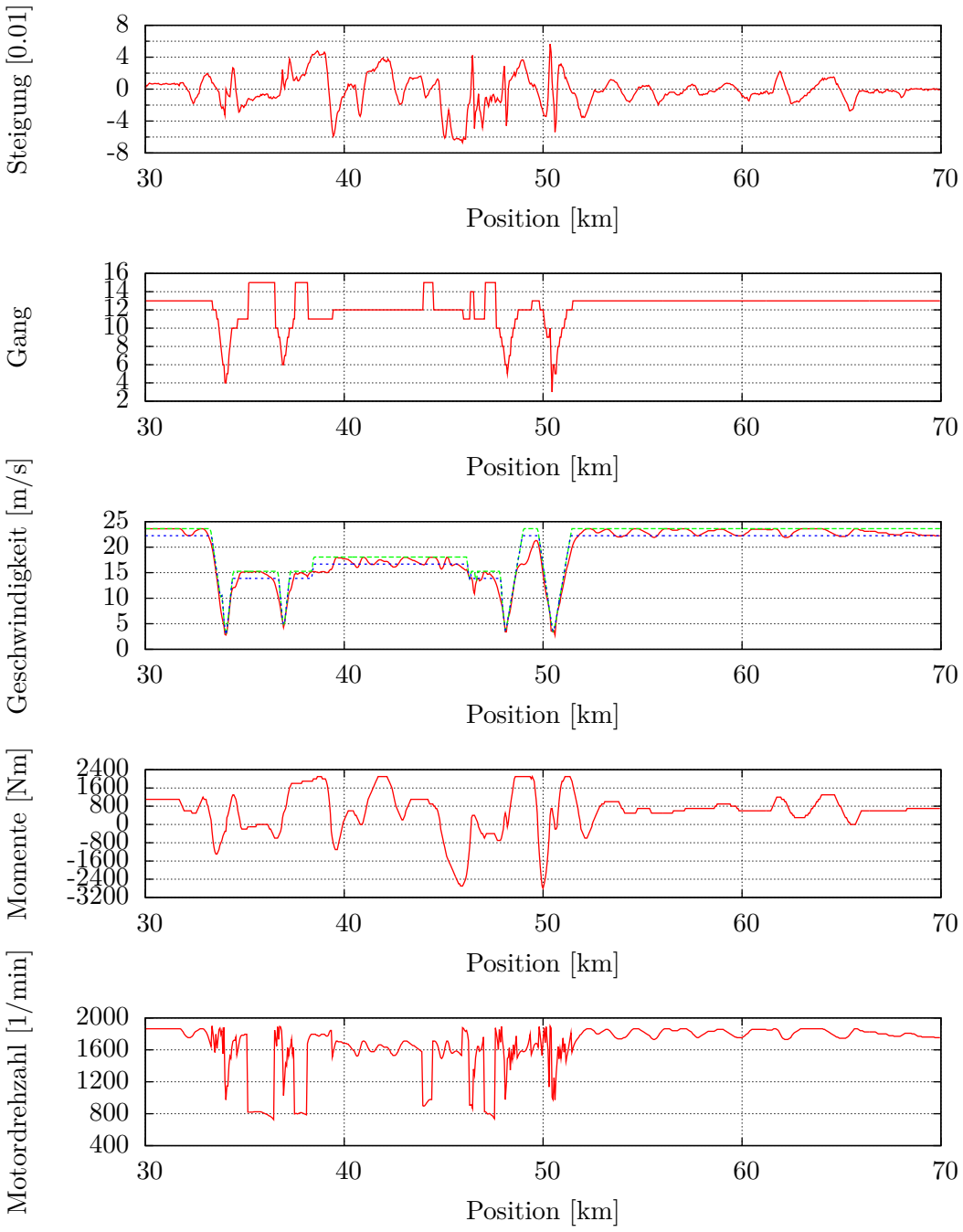

Abbildung 7.13: Optimalsteuerung bei einer Fahrzeugmasse von 42 t. $\delta_s = 50$ m, $\delta_v = 0.1\frac{\mathrm{m}}{\mathrm{s}}$, $\delta_M = 100$ Nm.

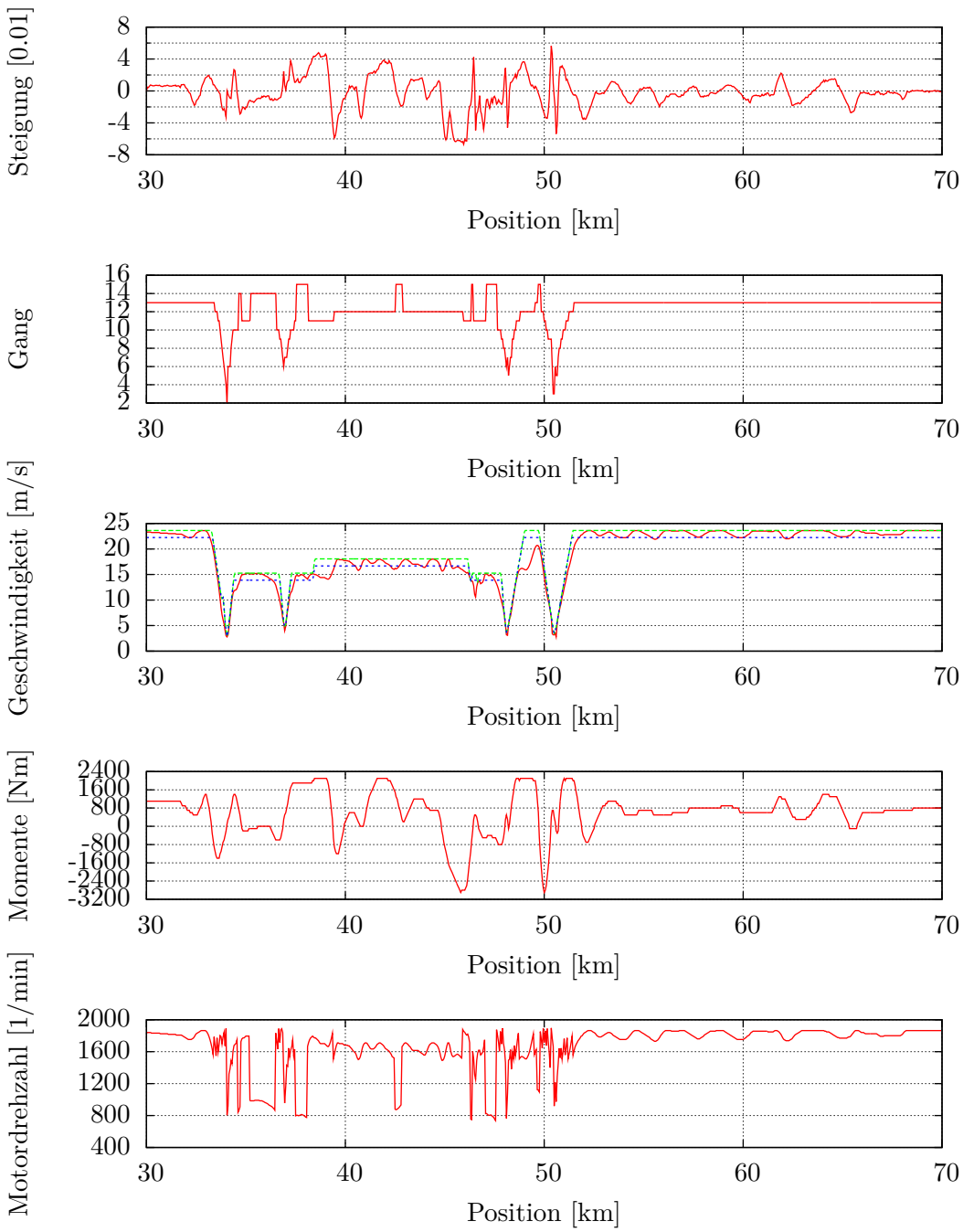

Abbildung 7.14: Optimalsteuerung bei einer Fahrzeugmasse von 45 t. $\delta_s = 50$ m, $\delta_v = 0.1\frac{\mathrm{m}}{\mathrm{s}}$, $\delta_M = 100$ Nm.

Masse [t]	Zielfunktionswert	Abbildung
30	1076.200908	7.11
35	1431.372252	7.12
40	1892.324674	7.3
42	2148.214550	7.13
45	2427.136142	7.14

Tabelle 7.2: Übersicht der Zielfunktionswerte für verschiedene Massen.

der Fall ist, sondern bleibt für eine Strecke von ungefähr einem Kilometer Länge darunter, obwohl in den 12. Gang heruntergeschaltet und dort mit hoher Drehzahl gefahren wird.

Ein ähnliches Phänomen ist zwischen Kilometer 49 und 50 zu entdecken. Auch hier bleibt die Lösungstrajektorie – zum Teil deutlich – unterhalb der Wunschgeschwindigkeit. Hier scheint der Treibstoffverbrauch zum Erreichen der Wunschgeschwindigkeit höher zu sein als die Penalisierung der Unterschreitung eben dieser Geschwindigkeit. Zusätzlich spielt an dieser Stelle sicherlich eine Rolle, dass die Maximalgeschwindigkeit kurz danach sehr stark abfällt und der 45 t schwere LKW stark gebremst werden muss. In der schwereren Variante muss mit einem Bremsmoment von über 3000 Nm gesteuert werden, um die Geschwindigkeit zu drosseln. Beim leichterem LKW fällt die Änderung der Momente nicht so stark aus, da es schlicht nicht so viel Bremskraft erfordert, eine geringere Masse zu bremsen.

Auf dem Abschnitt zwischen Kilometer 40 und 44 liegt beim 45 t schweren LKW durchgängig ein höheres Motormoment vor als beim wesentlich leichter beladenen Vergleichs-LKW mit 30 t Gewicht, was dann auch zu einem erhöhten Treibstoffverbrauch führt.

Trotz der Ähnlichkeiten der optimalen Trajektorien unterscheiden sich die Zielfunktionswerte aufgrund der veränderten Fahrzeugdynamik teilweise stark.

Ein Vergleich der Zielfunktionswerte der Optimalsteuerungen für verschiedene Massen ist in Tabelle 7.2 zu finden.

7.4.2 Wind

In diesem Abschnitt beschreiben wir den Einfluss von Gegenwind auf die Ergebnisse der *Dynamischen Programmierung*.

Um den Gegenwind zu modellieren, greifen wir auf die Gleichung für den *Strömungswiderstand* (5.5) aus Kapitel 5 zurück, welche wir hier kurz wiederholen wollen:

$$M_{\text{air}}(s) := \frac{1}{2}\, c_{\text{w}}\, A\, \rho_{\text{air}}\, v^2(s). \tag{7.1}$$

Gegenwind $[\frac{\text{km}}{\text{h}}]$	Zielfunktionswert	Abbildung
0	1892.324674	7.3
10	1933.465501	7.15
30	2036.969712	7.16
50	2129.963699	7.17
80	2358.359558	7.18

Tabelle 7.3: Übersicht der Zielfunktionswerte für verschieden starken Gegenwind.

Diese ändern wir zu

$$M_{\text{air}}(s) := \frac{1}{2}\, c_{\text{w}}\, A\, \rho_{\text{air}}\, (v + v_{\text{wind}})^2(s) \tag{7.2}$$

und halten v_{wind} über die gesamte Strecke konstant. Durch die von uns gewählte Art der Modifizierung von (7.1) beschränken wir uns auf die Annahme, dass der Wind durchgehend frontal auf die Frontfläche des LKW trifft.

Der erhöhte Strömungswiderstand führt dazu, dass bei unveränderter Steuerung die Beschleunigung verringert wird. Daher sind auch die in den Abbildungen 7.15 bis 7.18 zu sehenden Effekte leicht nachzuvollziehen.

Mit steigenden Windgeschwindigkeiten hebt sich auch über die gesamte Strecke durchgehend das Grundniveau der Momente, um dem steigenden Widerstand entgegenzusteuern. Bei Streckenabschnitten, an denen bereits das drehzahlabhängige Maximum des Motormoments abgerufen wird, führt der Gegenwind zwangsläufig dazu, dass die Wunschgeschwindigkeit unterschritten wird, da die für die Aufrechterhaltung der Geschwindigkeit nötige Beschleunigung des LKW nicht realisiert werden kann. Dieses Phänomen wird natürlich besonders an den Stellen deutlich, an denen eine Steigung zu bewältigen ist.

Interessanterweise sind bei starkem Gegenwind auch zunehmend mehr Schaltvorgänge zu beobachten. Allerdings wird hier nicht – wie man vielleicht erwarten würde – gegenüber der Referenzlösung ohne Windeinfluss heruntergeschaltet. Es sind vielmehr einige Phasen zu erkennen, in denen zeitweise in einen höheren Gang gewechselt wird. Vermutlich ist das dem Umstand geschuldet, dass der Treibstoffverbrauch durch das höhere Motormoment gegenüber den anderen Komponenten der Zielfunktion an Bedeutung gewinnt. Durch das Schalten in einen höheren Gang wird die Drehzahl reduziert, was zur Folge hat, dass der Treibstoffverbrauch gesenkt wird.[25]

[25] Der Treibstoffverbrauch steigt sowohl mit dem Motormoment als auch mit der Motordrehzahl, siehe Abbildung 5.2.

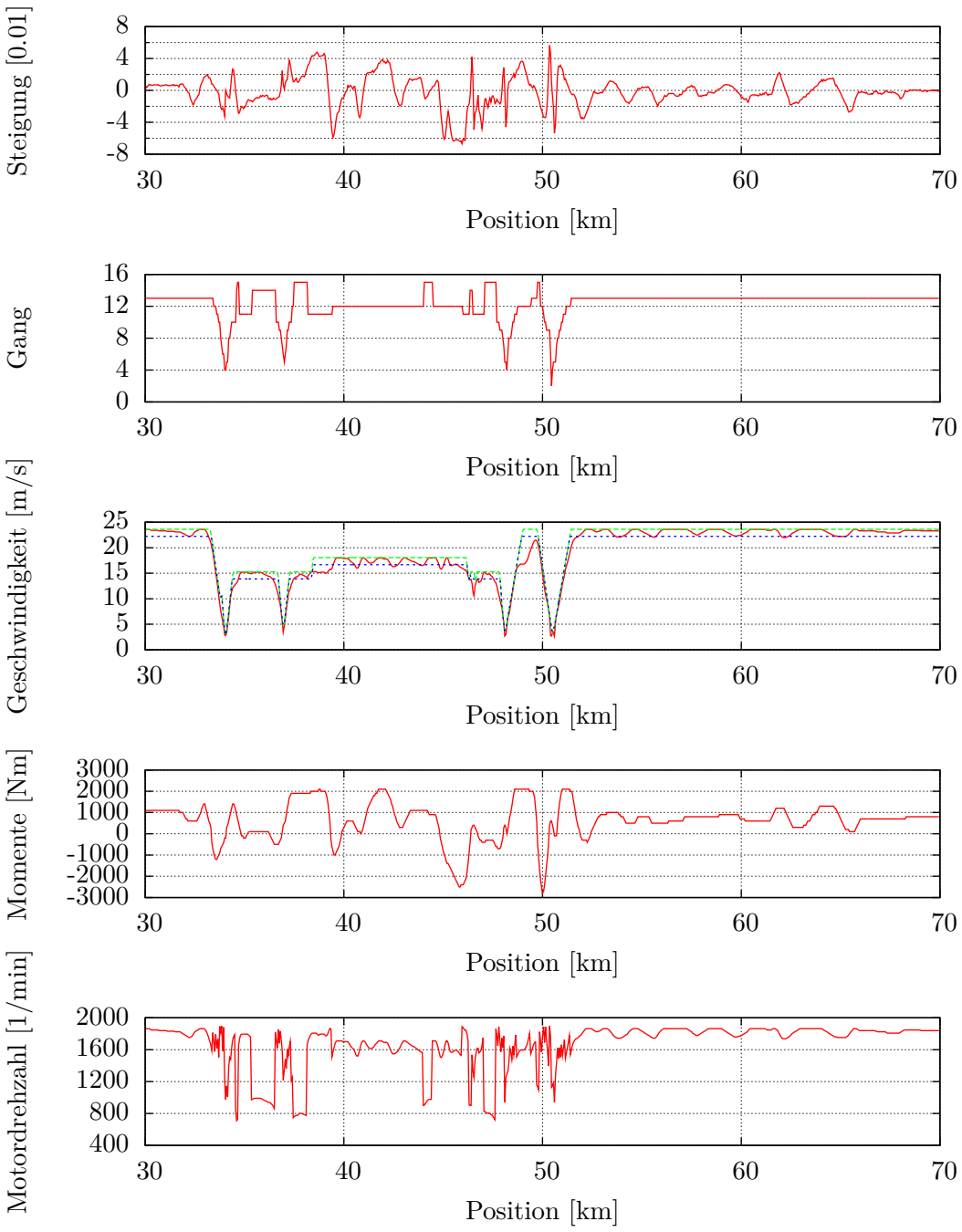

Abbildung 7.15: Optimalsteuerung bei 10 $\frac{\text{km}}{\text{h}}$ Gegenwind. $\delta_s = 50$ m, $\delta_v = 0.1\frac{\text{m}}{\text{s}}$, $\delta_M = 100$ Nm.

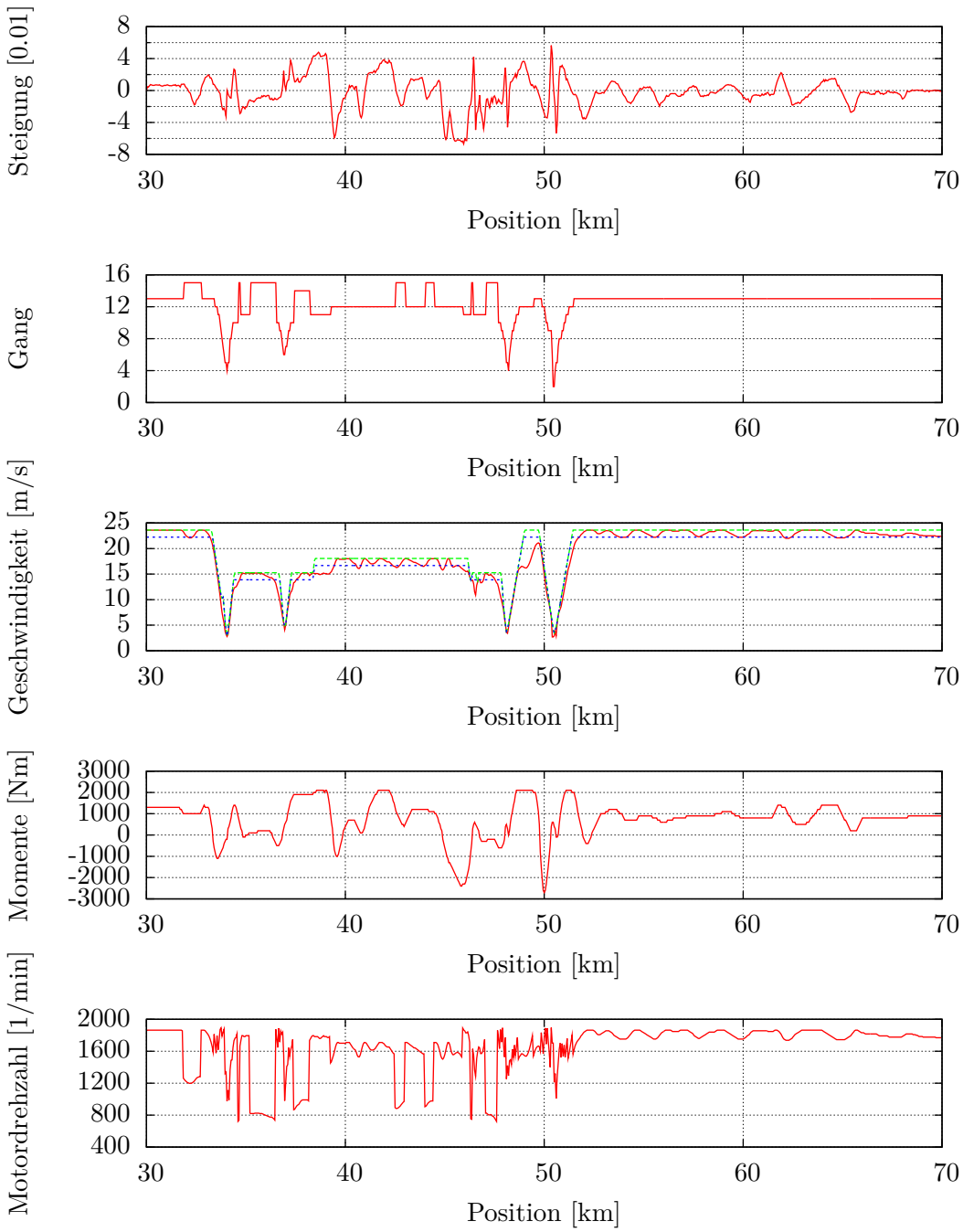

Abbildung 7.16: Optimalsteuerung bei 30 $\frac{\text{km}}{\text{h}}$ Gegenwind. $\delta_s = 50$ m, $\delta_v = 0.1\frac{\text{m}}{\text{s}}$, $\delta_M = 100$ Nm.

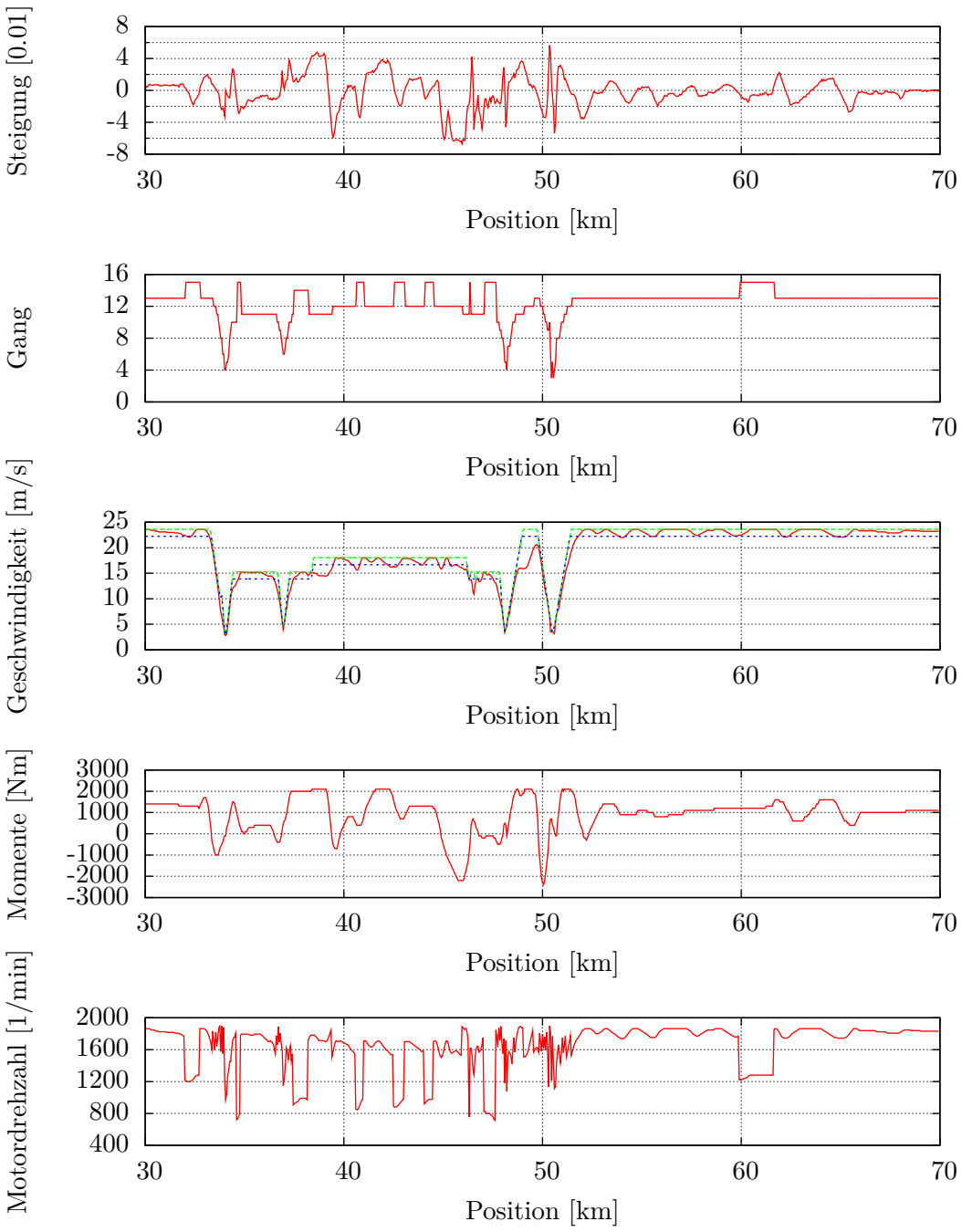

Abbildung 7.17: Optimalsteuerung bei 50 $\frac{\text{km}}{\text{h}}$ Gegenwind. $\delta_s = 50$ m, $\delta_v = 0.1\frac{\text{m}}{\text{s}}$, $\delta_M = 100$ Nm.

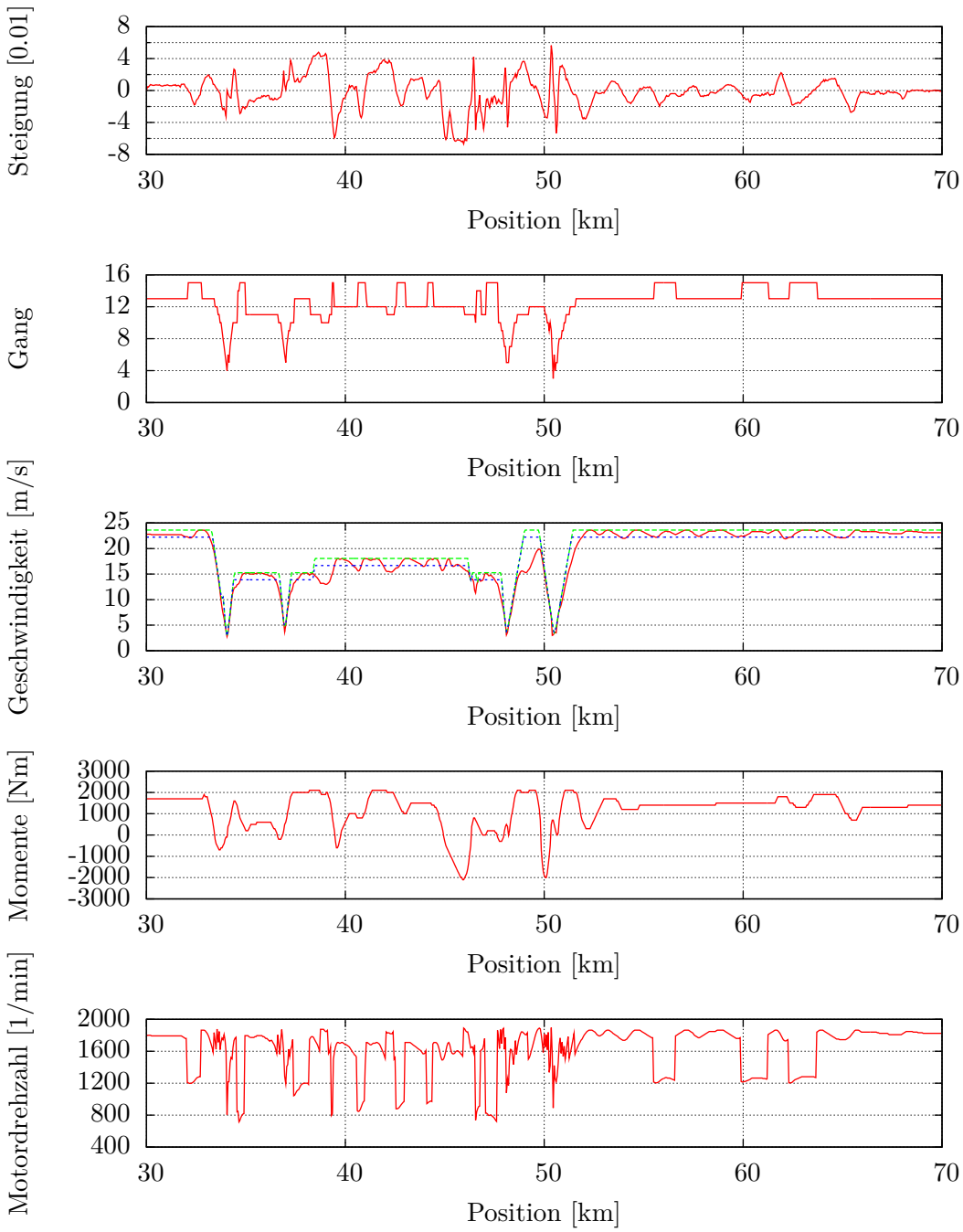

Abbildung 7.18: Optimalsteuerung bei 80 $\frac{\text{km}}{\text{h}}$ Gegenwind. $\delta_s = 50$ m, $\delta_v = 0.1\frac{\text{m}}{\text{s}}$, $\delta_M = 100$ Nm.

7.4.3 Einfluss der Diskretisierungparameter

Nun wollen wir untersuchen, welchen Einfluss die Veränderung der Diskretisierungsgrößen δ_s, δ_M und δ_v auf die Beschaffenheit der Lösung hat. Aus der Theorie[26] wissen wir bereits, dass die Laufzeit linear mit der Anzahl der Diskretisierungspunkte wächst und es daher den Konflikt zwischen langer Laufzeit und guter Approximation der eigentlich reellwertigen Größe gibt.

Diskretisierung des Ortes

Die Wahl von δ_s zur Diskretisierung des Ortes beeinflusst mehrere Faktoren. Zum einen werden in unserer Implementierung der *Dynamischen Programmierung* die Streckendaten lediglich an diesen Stellen ausgewertet. Bei zu grober Diskretisierung kann es also vorkommen, dass kritische Stellen des Streckenprofils „übersehen“ werden, da sie zwischen zwei Abtastpunkten liegen.

Zum anderen wird die Information der Steigung, die am Gitterpunkt τ_i ausgewertet wird, als konstant für das folgende Diskretisierungsintervall angenommen. Bei der Lösung der Differentialgleichung zur Bestimmung der neuen Geschwindigkeit wird dem numerischen Integrator dieser Wert übergeben und die gesamte Integration über die Strecke von δ_s Metern bis zum nächsten Diskretisierungspunkt τ_{i+1} wird mit diesem konstanten Wert durchgeführt.

Weiterhin ist an die örtliche Diskretisierung auch die Frequenz geknüpft, mit der die Steuerung geändert werden kann. Bei feiner Diskretisierung ist es also möglich, häufiger kleine Änderungen der Steuerungen vorzunehmen.

Neben unserer Standardortsdiskretisierung von $\delta_s = 50$ m findet man in den Abbildungen 7.19 und 7.20 Rechnungen für 25 bzw. 100 m Abstand zwischen zwei Gitterpunkten.

Beim Vergleich von $\delta_s = 25$ m und $\delta_s = 50$ m fällt auf den ersten Blick auf, dass in der feiner diskretisierten Optimalsteuerung auch von der Möglichkeit, häufiger schalten zu können, Gebrauch gemacht wird. Dadurch gelingt es der feineren Steuerung auch besser, die Wunschgeschwindigkeiten zu halten und insgesamt auch den Zielfunktionswert deutlich auf 1651.016443 zu senken.

Erhöht man δ_s auf 100 m, ergeben sich ähnlich Effekte wie bei der in Abschnitt 7.2 untersuchten Verkürzung des Prädiktionshorizonts. Kurz nach Streckenkilometer 50, wo Wunsch- und Maximalgeschwindigkeit erst stark fallen und dann schnell wieder steigen, ist die zu grobe Diskretisierung wahrnehmbar. Es wird der Tiefpunkt dieser beiden Geschwindigkeitsvorgaben „verpasst“, da in diesem Szenario nur noch alle 100 m die Steuerung geändert werden kann. So ist die gefahrene Geschwindigkeit fast den ganzen Abstieg und den ganzen Wiederanstieg deutlich niedriger als die Wunschgeschwindigkeit.

Auch ist es der Optimalsteuerung in diesem Fall nicht möglich, auf Streckenabschnitten konstanter Wunsch- und Maximalgeschwindigkeit – so z.B. Streckenkilometer 38 bis

[26] Siehe Abschnitt 2.3.5.

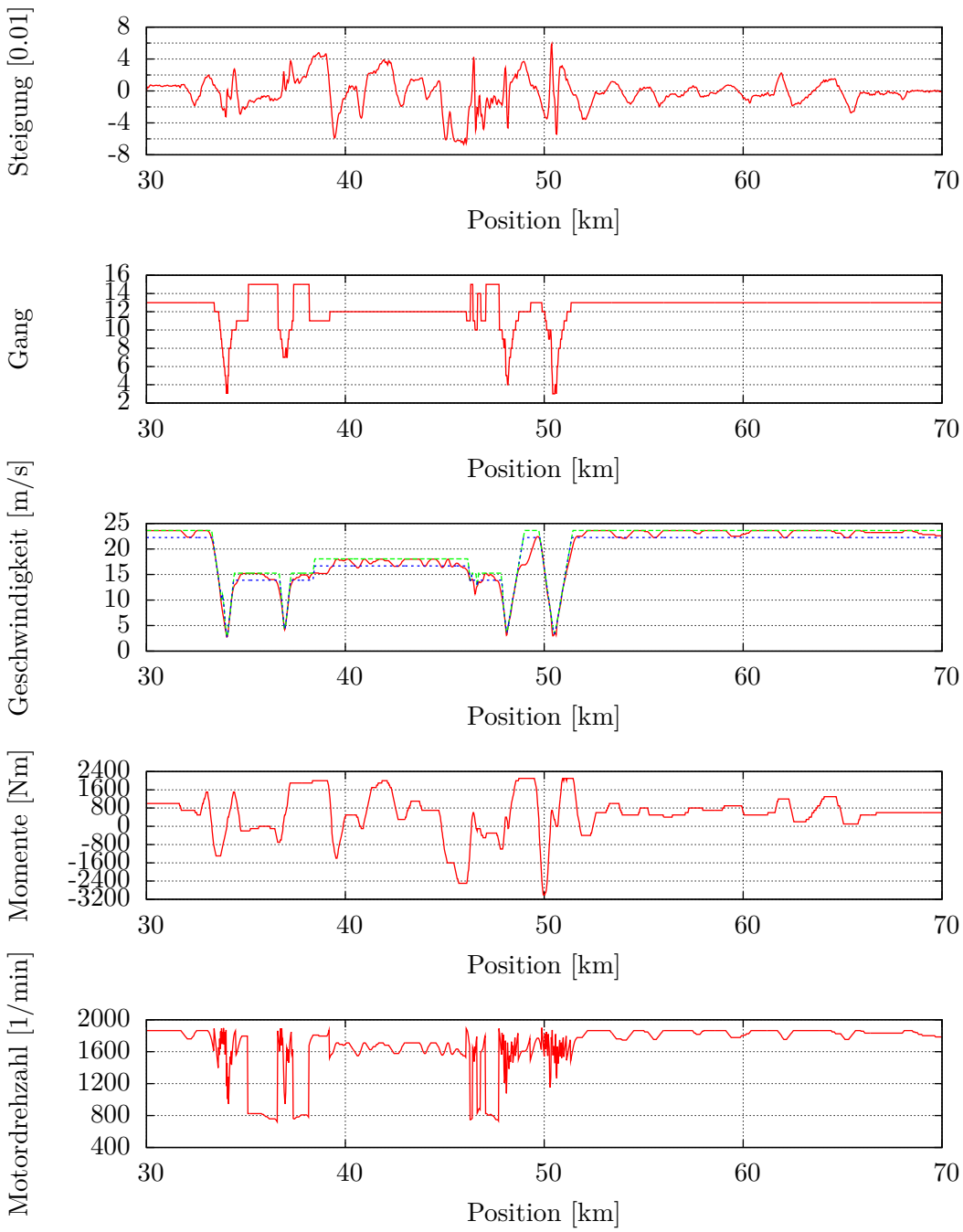

Abbildung 7.19: Optimalsteuerung für reduziertes δ_s. $\delta_s = 25$ m, $\delta_v = 0.1\frac{\mathrm{m}}{\mathrm{s}}$, $\delta_M = 100$ Nm.

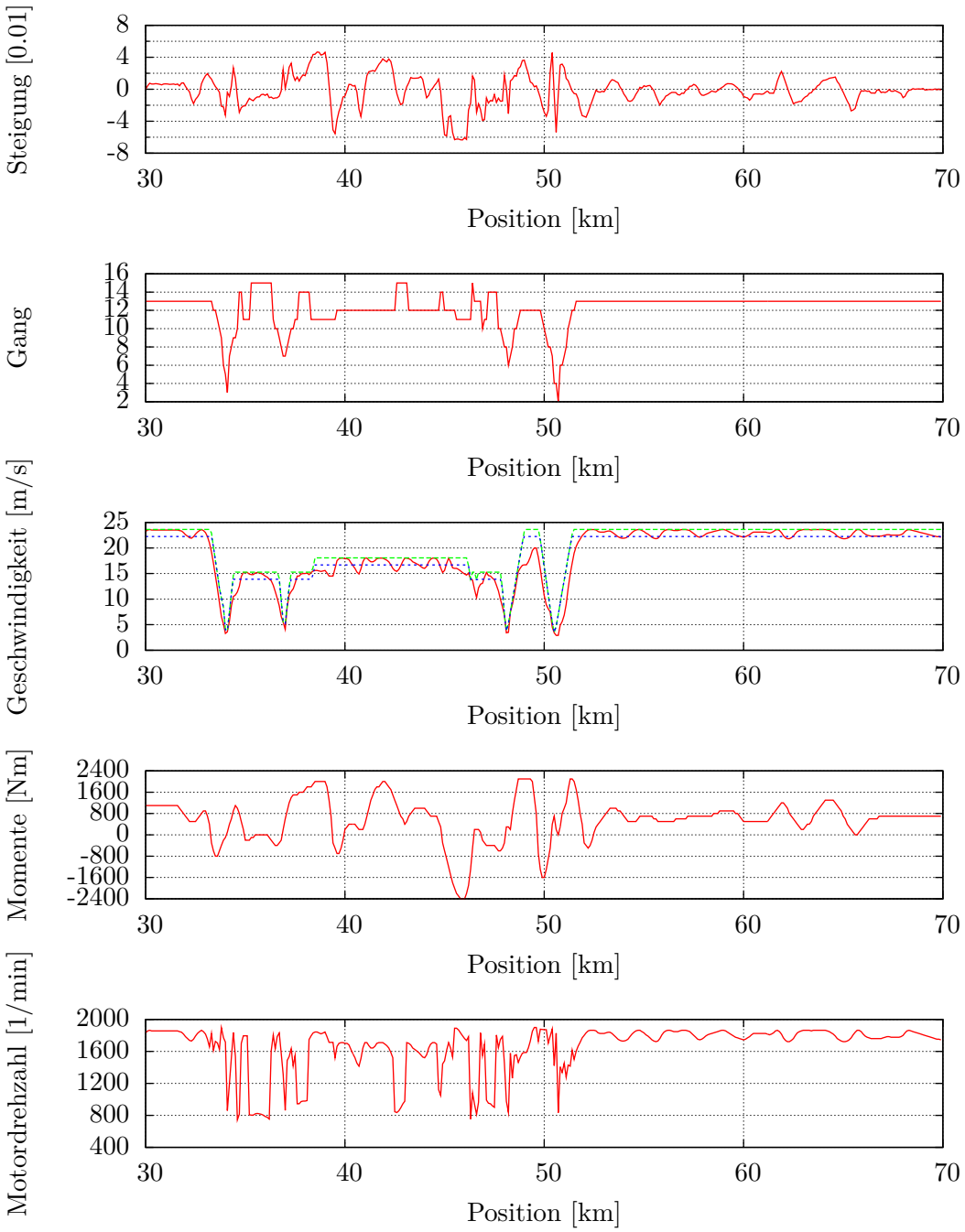

Abbildung 7.20: Optimalsteuerung für erhöhtes δ_s. $\delta_s = 100$ m, $\delta_v = 0.1\frac{\mathrm{m}}{\mathrm{s}}$, $\delta_M = 100$ Nm.

δ_s [m]	Zielfunktionswert	Abbildung
25	1651.016443	7.19
50	1892.324674	7.3
100	2393.693134	7.20

Tabelle 7.4: Übersicht der Zielfunktionswerte für verschiedene Ortsdiskretisierungen.

δ_v $[\frac{\mathsf{m}}{\mathsf{s}}]$	Zielfunktionswert	Abbildung
0.05	1884.884133	7.21
0.1	1892.324674	7.3
0.5	1879.491795	7.22

Tabelle 7.5: Übersicht der Zielfunktionswerte für verschiedene Diskretisierungen der Geschwindigkeit.

46 – die Geschwindigkeit im Bereich zwischen diesen beiden Geschwindigkeitsvorgaben zu halten. Immer wieder fällt die Geschwindigkeit – eigentlich ohne erkennbare Notwendigkeit – unter die Wunschgeschwindigkeit[27] und erhöht so den Zielfunktionswert, siehe Tabelle 7.4.

Man sieht also deutlich, dass durch eine zu grobe Diskretisierung des Ortes Probleme entstehen, die vermeidbar sind. Außerdem können für diese Probleme sowohl die zu grobe Ortsdiskretisierung als auch eine Verkürzung des Prädiktionshorizonts verantwortlich gemacht werden, was zu Fehlinterpretationen der Ursache führen kann.

Diskretisierung der Geschwindigkeit

Die nächste Größe, die für die *Dynamische Programmierung* diskretisiert werden muss, ist die Geschwindigkeit, die gleichzeitig eine der beiden zu diskretisierenden Zustände in unserem Modell darstellt.

Die Effekte, die bei der Wahl von δ_v Beachtung finden sollten, sind weniger solche, die sich an der Zielfunktion messen lassen können (Tabelle 7.5), sondern eher von der Art, dass sie qualitative Aspekte beeinflussen.

Ein naheliegender, aber nicht zu vergessender Aspekt ist, dass durch eine grobe Diskretisierung der Geschwindigkeit die Grenzen der Maximal- und Wunschgeschwindigkeit praktisch verschoben werden. So wird eine Grenze, die reellwertig bei einem beispielhaften Wert von 22.8 $\frac{\mathsf{m}}{\mathsf{s}}$ liegt, bei einer Diskretisierung von $\delta_v = 0.5\frac{\mathsf{m}}{\mathsf{s}}$ auf den nächstkleineren Wert des Diskretisierungsgitters – etwa 22.5 $\frac{\mathsf{m}}{\mathsf{s}}$ – „verschoben“.

[27] Im Gegensatz zur Wunschgeschwindigkeit ist die Maximalgeschwindigkeit eine *harte* Grenze, die nicht – auch nicht temporär – überschritten werden darf, siehe Kapitel 5.

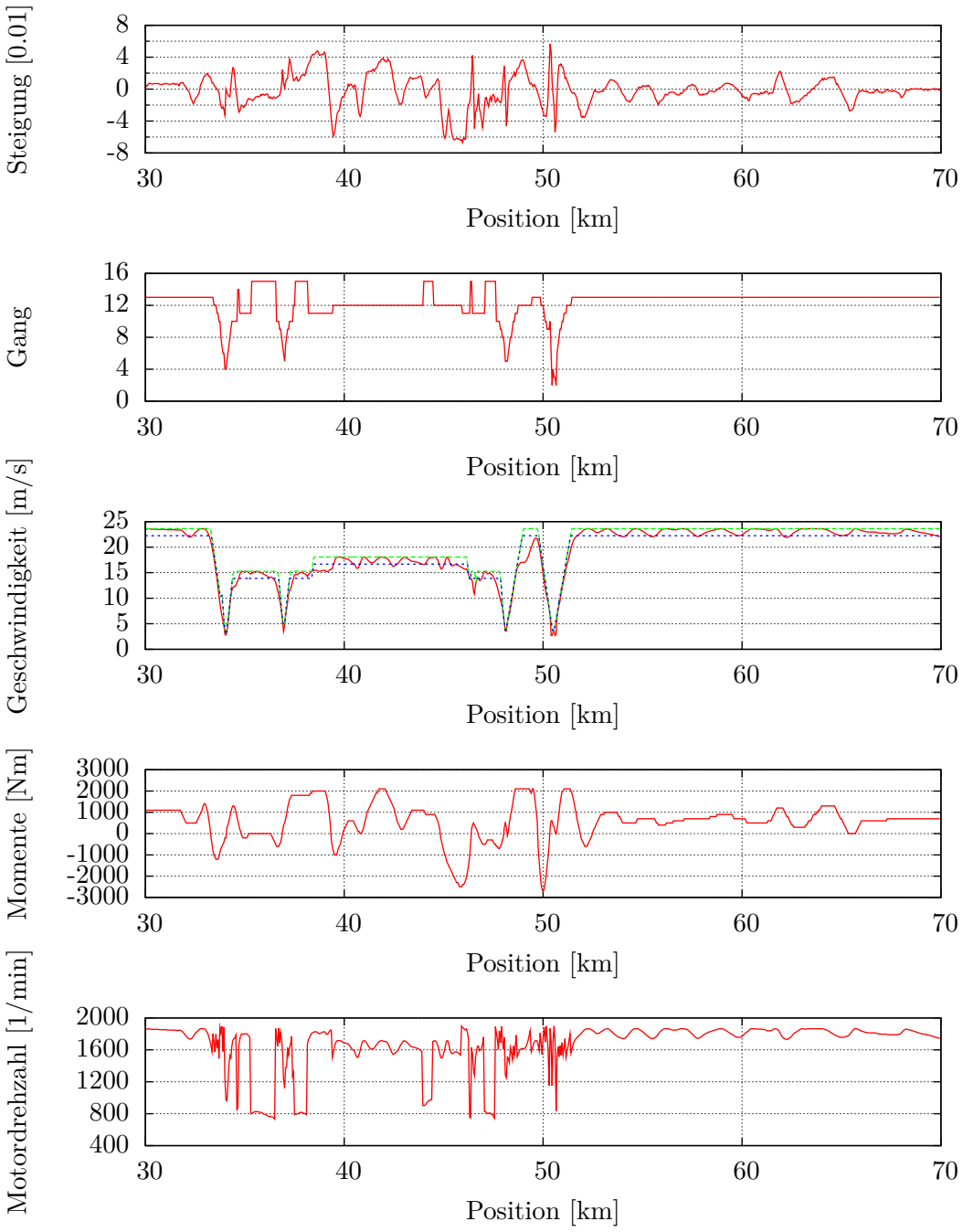

Abbildung 7.21: Optimalsteuerung für reduziertes δ_v. $\delta_s = 50$ m, $\delta_v = 0.05\frac{\mathrm{m}}{\mathrm{s}}$, $\delta_M = 100$ Nm.

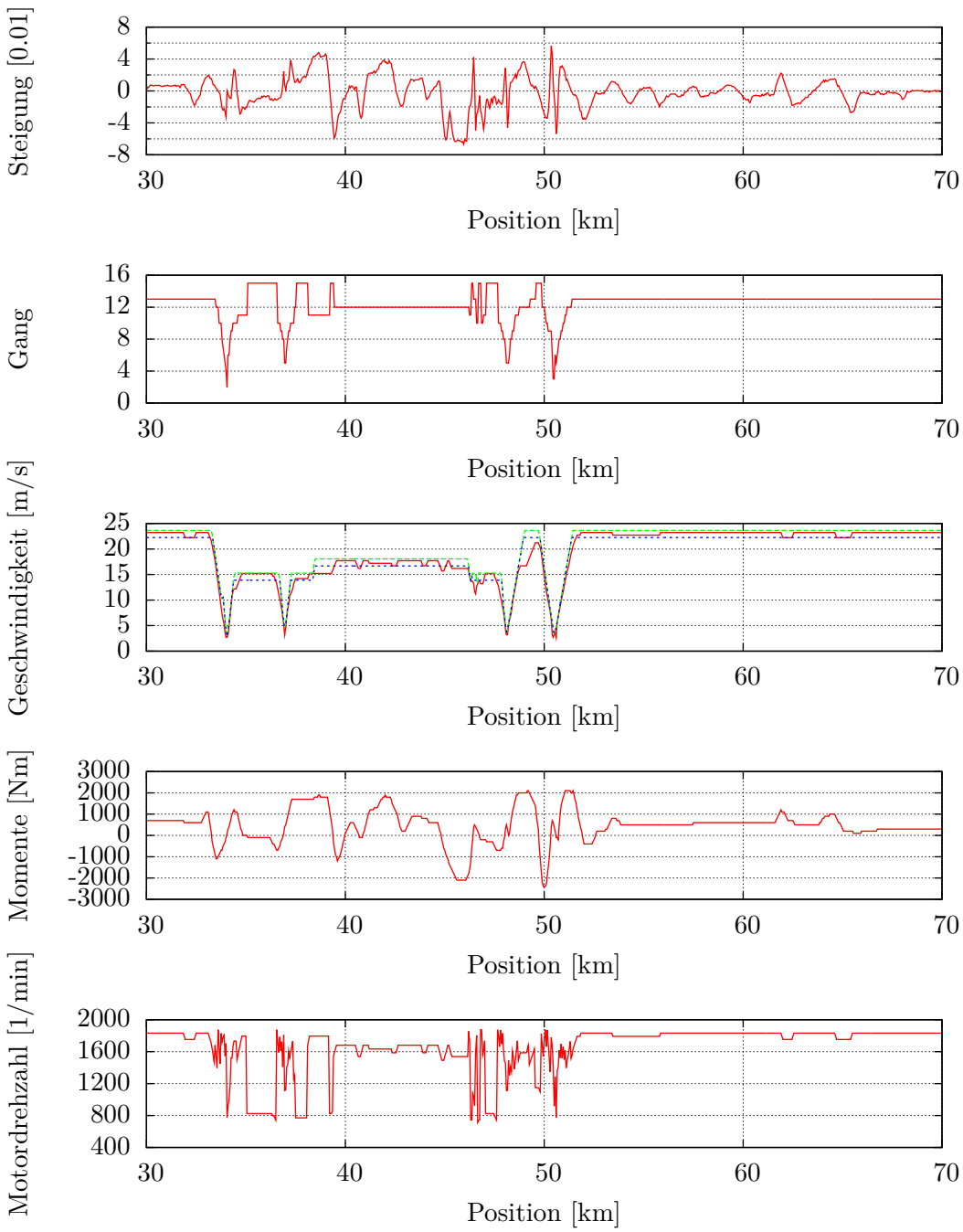

Abbildung 7.22: Optimalsteuerung für erhöhtes δ_v. $\delta_s = 50$ m, $\delta_v = 0.5\frac{\mathrm{m}}{\mathrm{s}}$, $\delta_M = 100$ Nm.

δ_M [Nm]	Zielfunktionswert	Abbildung
20	1714.441152	7.23
50	1761.951097	7.24
100	1892.324674	7.3
150	2015.031832	7.25
200	2154.796518	7.26

Tabelle 7.6: Übersicht der Zielfunktionswerte für verschiedene Diskretisierungen der Momente.

Ein weiterer Gesichtspunkt, den man bei der Diskretisierung der Geschwindigkeit bedenken sollte, ist die Übergangsfunktion $x_{k+1} = f_k(x_k, u_k)$ für die Geschwindigkeit. Wie bereits in Kapitel 6 erwähnt, wird das Ergebnis von f_k durch numerische Integration der Geschwindigkeitsdifferentialgleichung bestimmt. Anschließend muss das Ergebnis noch auf einen Gitterpunkt der Geschwindigkeitsdiskretisierung gerundet werden, damit beispielsweise Fortsetzungskosten für v_{k+1} nachgeschlagen werden können. Das Ergebnis des Rundens kann hierbei im schlimmsten Fall $\frac{1}{2}\delta_v$ vom Ergebnis der Integration entfernt liegen.

Hier entsteht also ein Diskretisierungsfehler, der die Dynamik des Modells verändert. Dies kann beispielsweise zu einer verfälschten Steuerung u führen, wenn das Ergebnis der Integration – bei Geschwindigkeit v startend – mit dieser Steuerung $v - \frac{1}{2}\delta_v + \epsilon,\ \epsilon > 0$ ist. In diesem Fall wird das Ergebnis auf v aufgerundet, die Geschwindigkeit kann also gehalten werden. Dabei ist das Motormoment u^*, das eigentlich benötigt wird, um die Geschwindigkeit zu halten, in Wirklichkeit größer. Da das kleinere Moment einen geringeren Treibstoffverbrauch verursacht, wird dieses aber gegenüber dem richtigen Wert bevorzugt. Da bei Steuerung mit u^* ebenfalls die Geschwindigkeit v gehalten wird, sieht man an diesem Beispiel, dass es passieren kann, dass verschiedene Steuerungen zum gleichen neuen Zustand führen können. Es sollte durch geeignete Wahl von δ_v versucht werden, die Auswirkungen dieses Effekts möglichst gering zu halten.

Ein wesentlicher Unterschied der Geschwindigkeitstrajektorien zwischen den Optimalsteuerungen für verschiedene Werte für δ_v ist nicht zu erkennen. Allerdings ist das eben beschriebene Phänomen der bei gröberen Geschwindigkeitsdiskretisierungen verfälschten Momente bei $\delta_v = 0.5\frac{\mathsf{m}}{\mathsf{s}}$ zu beobachten. Gerade auf dem Teilstück nach Kilometer 50 bleibt die Geschwindigkeit nahezu konstant, obwohl im Vergleich zu den anderen Optimalsteuerungen mit einem niedrigeren Motormoment gesteuert wird.

Diskretisierung der Momente

In diesem Abschnitt wollen wir kurz die Diskretisierung der eigentlich kontinuierlichen Steuerung der Momente untersuchen. In den Abbildungen 7.23 bis 7.26 sind Plots der

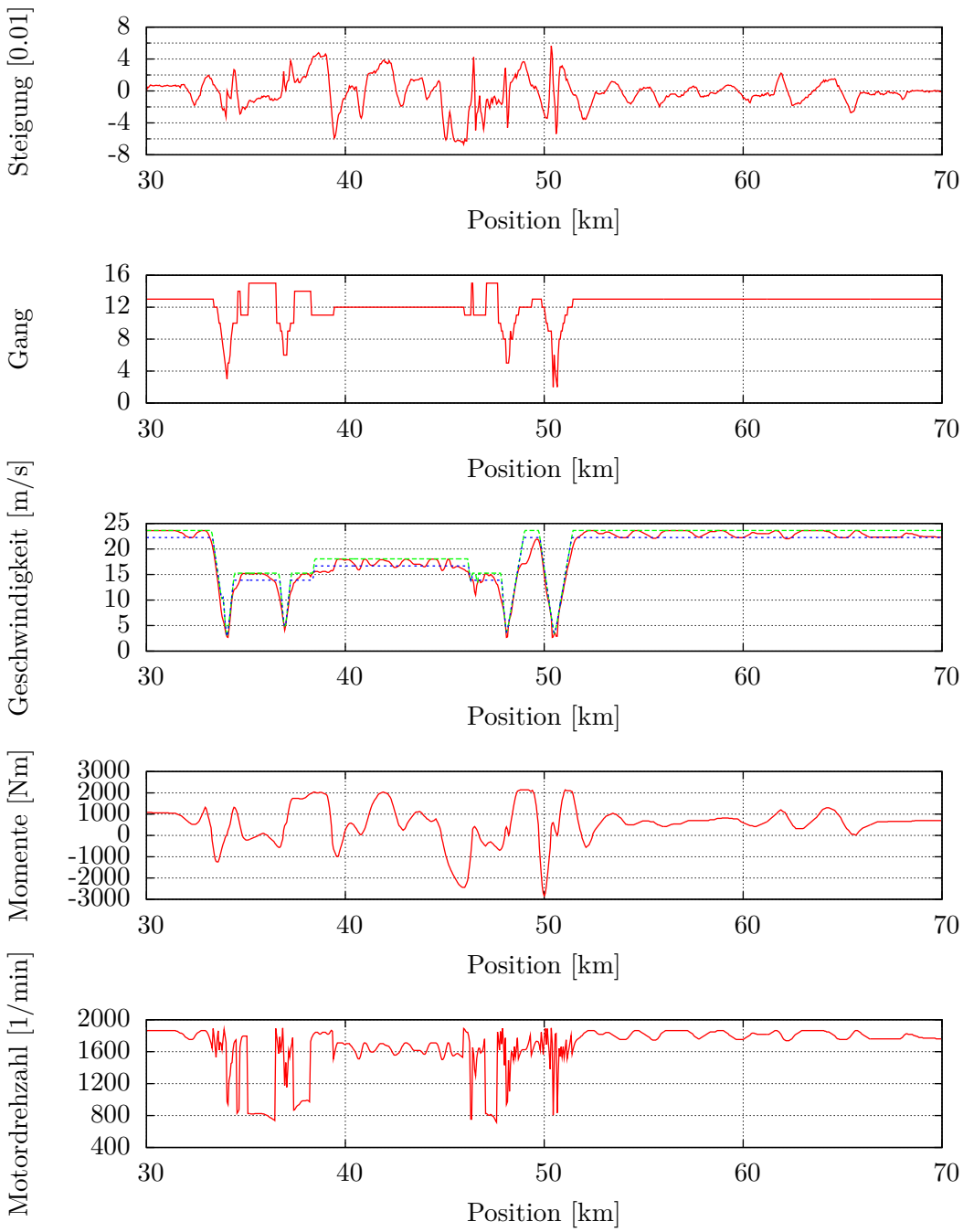

Abbildung 7.23: Optimalsteuerung für reduziertes δ_M. $\delta_s = 50$ m, $\delta_v = 0.1\frac{\mathrm{m}}{\mathrm{s}}$, $\delta_M = 20$ Nm.

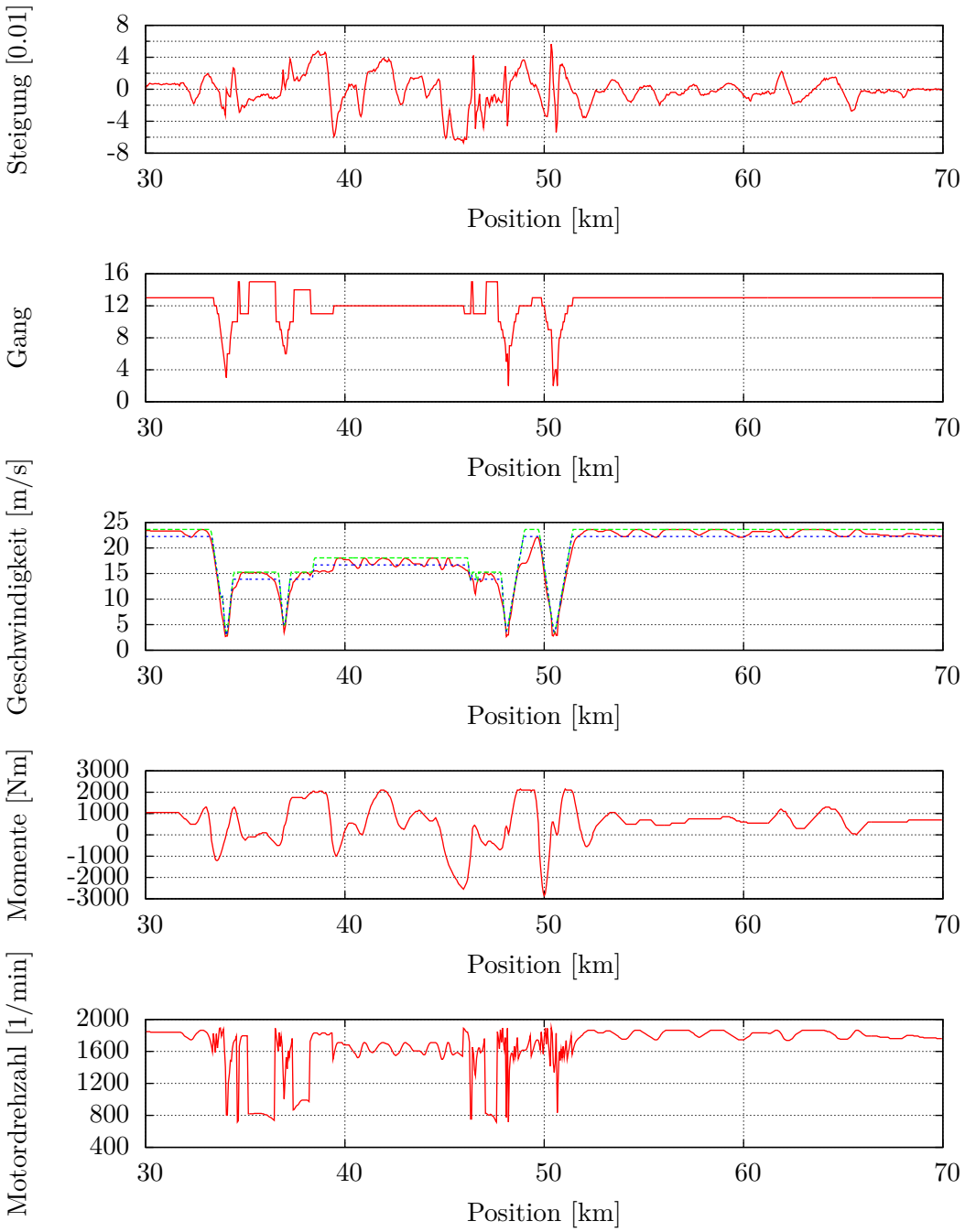

Abbildung 7.24: Optimalsteuerung für reduziertes δ_M. $\delta_s = 50$ m, $\delta_v = 0.1\frac{\mathrm{m}}{\mathrm{s}}$, $\delta_M = 50$ Nm.

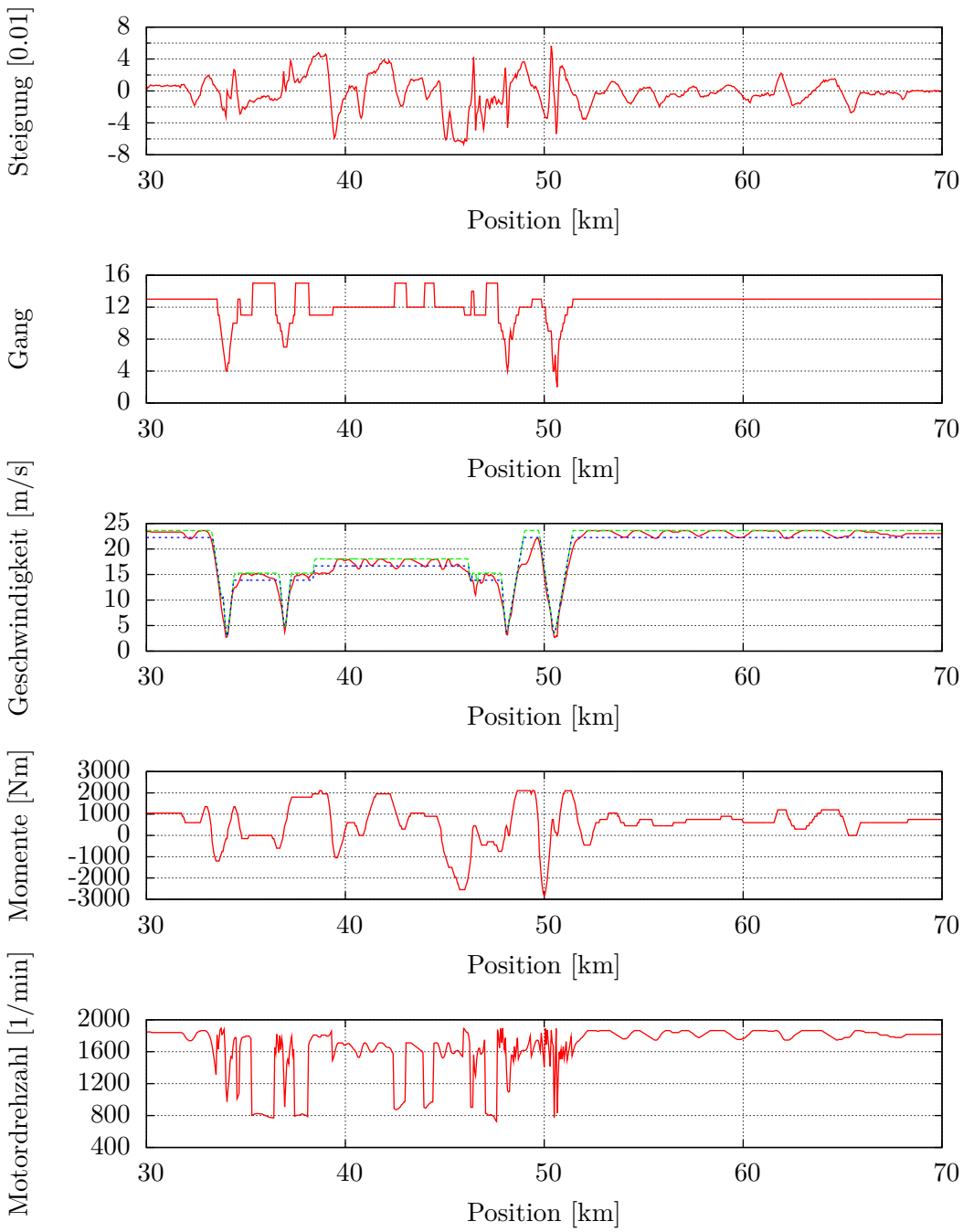

Abbildung 7.25: Optimalsteuerung für erhöhtes δ_M. $\delta_s = 50$ m, $\delta_v = 0.1\frac{\text{m}}{\text{s}}$, $\delta_M = 150$ Nm.

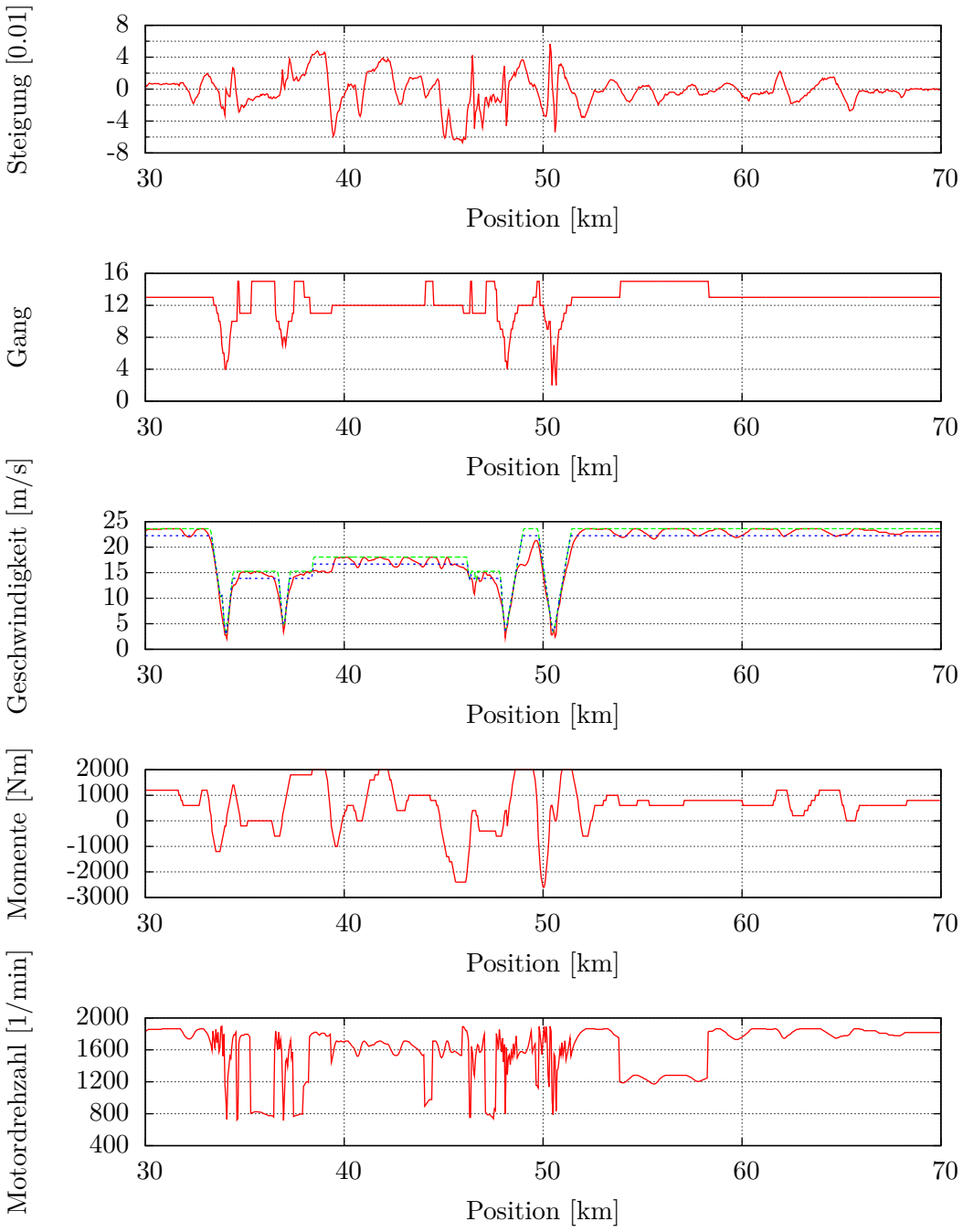

Abbildung 7.26: Optimalsteuerung für erhöhtes δ_M. $\delta_s = 50$ m, $\delta_v = 0.1\frac{\text{m}}{\text{s}}$, $\delta_M = 200$ Nm.

Ergebnisse mit verschiedenen Diskretisierungsgrößen dargestellt. Zusätzlich dient – wie in den vorherigen Abschnitten bereits mehrfach erwähnt – Abbildung 7.3 als Referenzlösung mit der Diskretisierung $\delta_M = 100$ Nm.

Betrachtet man die Ergebnisse, so ergeben sich auf den ersten Blick keine großen Unterschiede im Verlauf der Lösungstrajektorien. Selbst bei Vergleich der hier verwendeten Extrema $\delta_M = 20$ Nm und $\delta_M = 200$ Nm sind nur marginale Nichtübereinstimmungen zu erkennen. Allerdings ist die Kurve der Momente für eine Diskretisierung von 20 Nm auf der ganzen Länge der Strecke etwas glatter, was aber selbstverständlich sein sollte. Die Charakteristiken der Momentenverläufe sind dagegen die gleichen. Zum Beispiel ist ein rascher Wechsel von hohem Motormoment zu hohem Bremsmoment bei Streckenkilometer 50 unabhängig von der Feinheit der Diskretisierung zu erkennen.

Allerdings ist mit zunehmend gröberer Diskretisierung ein Anwachsen der Zielfunktion zu beobachten, welches aber nicht so stark ausfällt, wie bei den vorherigen Abhängigkeitsuntersuchungen anderer Größen. So ist bei einem fünf mal so feinen Gitter – gegenüber dem in dieser Arbeit als Standard benutzten Wert von $\delta_M = 100$ Nm – ein um etwa 10 % kleinerer Zielfunktionswert festzustellen. Die Verringerung des Wertes ist dabei in erster Linie nicht auf eine verbesserte Geschwindigkeitstrajektorie oder die Einsparung von Gangwechseln sondern auf den glatteren Verlauf der Momente selbst zurückzuführen. Da die Änderung der Momente quadratisch bestraft wird, ist es durch die feinere Diskretisierung möglich, eine Änderung, die im 100 Nm-Gitter durchgeführt wurde, in mehrere kleinere Änderungen zu splitten und dadurch den Zielfunktioswert zu verkleinern. Verdeutlicht werden kann dies an der einfachen Ungleichung

$$(\Delta M)^2 \geq n(\frac{1}{n}\Delta M)^2 = \frac{1}{n}(\Delta M)^2, \quad \forall n \in \mathbb{N}. \tag{7.3}$$

ΔM ist hier eine Änderung der Momente, die bei feinerer Diskretisierung in n Teiländerungen à $\frac{1}{n}\Delta M$ aufgeteilt wird.

Insgesamt kann also behauptet werden, dass die Feinheit der Diskretisierung der Momente einen eher kleinen Beitrag zur Verbesserung der Lösung leistet, wenn man davon absieht, dass feinere Änderungen in unserer Formulierung der Zielfunktion eigentlich bevorzugt werden.

7.4.4 Gewichtung der Zielfunktion

Zum Abschluss dieses Kapitels wollen wir Lösungen des Optimalsteuerungsproblems vorstellen, bei denen die Gewichte der Zielfunktionskomponenten variiert wurden, um ein anderes Verhalten des LKW zu erzeugen. Bei allen hier durchgeführten Rechnungen haben wir die *Dynamische Programmierung* mit den Diskretisierungen $\delta_s = 50$ m, $\delta_v = 0.1\frac{\mathrm{m}}{\mathrm{s}}$, $\delta_M = 100$ Nm benutzt. Wir werden hier die Beobachtungen erläutern, die qualitativ gemacht werden können. Eine Berechnung der *Pareto-Mengen* sollte in zukünftigen Arbeiten systematisch untersucht werden.

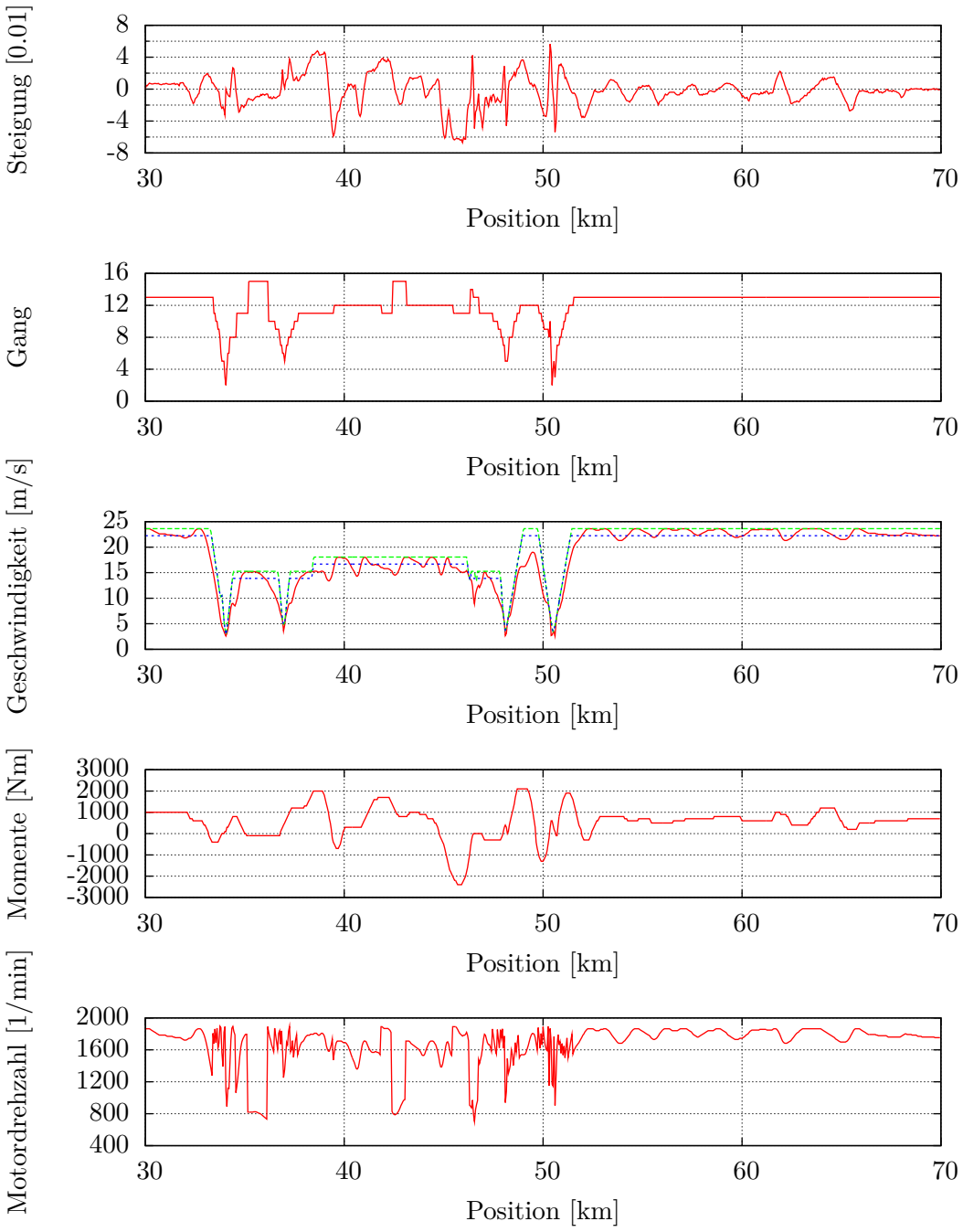

Abbildung 7.27: Optimalsteuerung für weniger Treibstoffverbrauch. $a_0 = 1.0$, $a_1 = 0.1$, $a_2 = 0.01$, $a_3 = 0.5$.

Gewicht	Zielfunktionskomponente
a_0	J_0: Treibstoffverbrauch
a_1	J_1: Unterschreitung der Wunschgeschwindigkeit
a_2	J_2: Änderung der Momente
a_3	J_3: Anzahl Schaltvorgänge

Tabelle 7.7: Zugehörigkeit der Gewichte a_i

Die vier verstellbaren Gewichte $a_i,\ i \in \{0, 1, 2, 3\}$ steuern den Einfluss, den die jeweilige Komponente auf die Gesamt-Zielfunktion hat, siehe Tabelle 7.7.

Da bei der in dieser Arbeit als Standard angenommenen Gewichtung $a_0 = 0.1$, $a_1 = 1.0$, $a_2 = 0.01$, $a_3 = 0.5$ die Wunschgeschwindigkeit nur sehr selten unterschritten wird[28], haben wir hier alternativ exemplarisch zwei Rechnungen mit Gewichtungen durchgeführt, die andere Zielfunktionskomponenten über die Einhaltung der Wunschgeschwindigkeit stellen.

In Abbildung 7.27 ist das Ergebnis einer Rechnung zu sehen, bei dem die Gewichte für den Treibstoffverbrauch (vorher 0.1) und die Einhaltung der Wunschgeschwindigkeit (vorher 1.0) vertauscht wurden. Die Gewichtung lautet also: $a_0 = 1.0$, $a_1 = 0.1$, $a_2 = 0.01$, $a_3 = 0.5$. Somit kommt dem Treibstoffverbrauch hier im Gegensatz zur Wunschgeschwindigkeit nun wesentlich mehr Bedeutung zu. Die anderen beiden Gewichte wurden nicht verändert.

Das gewünschte Verhalten ist auch in den Plots zu erkennen. Wenn man den Plot der Geschwindigkeit betrachtet, sieht man deutlich, dass die Minimalgeschwindigkeit nicht so streng wie vorher eingehalten wird. Typisch ist eine nennenswerte Stelle kurz vor Kilometer 50: Zwischen zwei Stellen, an denen die Maximalgeschwindigkeit aufgrund von Kurven stark beschränkt ist, existiert eine ungefähr einen Kilometer lange Strecke, auf der schnell gefahren werden kann. Im Gegensatz zu Lösung 7.3, in der zumindest durch starke Beschleunigung gefolgt von starkem Bremsen (Bremsmoment 3000 Nm) der Versuch deutlich wird, auch in diesem Abschnitt durchgängig die Wunschgeschwindigkeit zu halten, verzichtet die alternativ gewichtete Lösung auf zu starke Beschleunigung. In Letzterer wird eine Geschwindigkeitstrajektorie gewählt, die die Wunschgeschwindigkeit deutlicher unterschreitet, starkes Bremsen und somit auch die unnötige Umwandlung von Treibstoff in Wärme allerdings vermeidet. Eine ähnliche Situation ergibt sich auch bei Kilometer 33 bis 37.

Als weitere Alternative haben wir eine Rechnung mit den Gewichten $a_0 = 0.1$, $a_1 = 1.0$, $a_2 = 1.0$, $a_3 = 0.5$ durchgeführt. Das Gewicht für die Bestrafung der Momentänderungen wurde also sehr stark um den Faktor 100 angehoben. Diese Maßnahme zielt auf einen Komfortgewinn des Fahrers ab. Die restlichen Gewichte bleiben gegenüber unserem Standard unverändert.

[28] Siehe Abbildung 7.3.

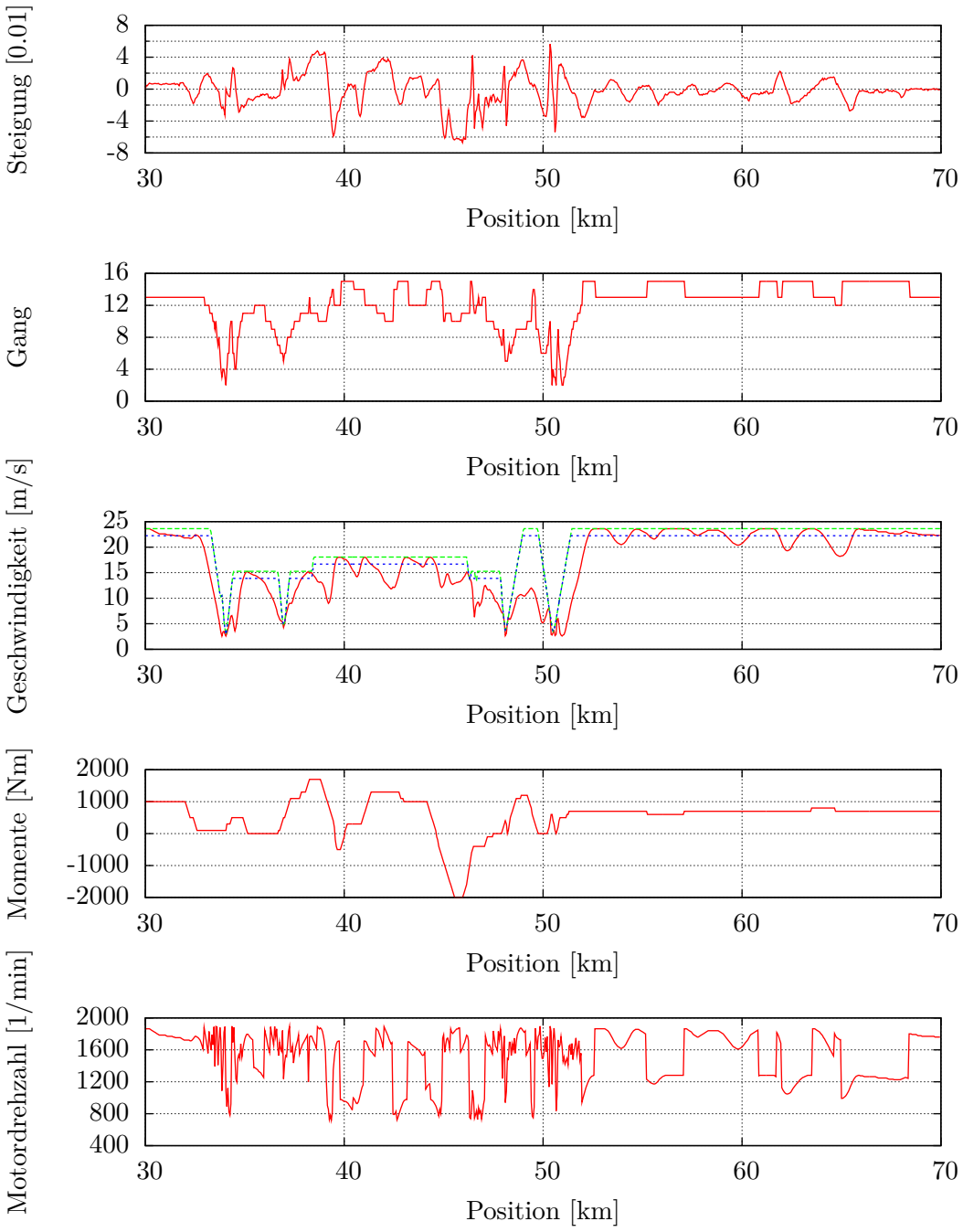

Abbildung 7.28: Optimalsteuerung für mehr Komfort. $a_0 = 0.1$, $a_1 = 1.0$, $a_2 = 1.0$, $a_3 = 0.5$.

Wenn man die dazugehörige Abbildung 7.28 betrachtet, so fallen direkt mehrere Phänomene ins Auge. Zum einen ist hier sicherlich die Kurve der Momente zu erwähnen, die im Vergleich zu allen vorhergehenden Plots wesentlich mehr Abschnitte enthält, in denen die Momente unverändert auf einem konstanten Niveau gehalten werden. Auch das sonst in vielen Plots zu findende starke Bremsmoment von bis zu 3000 Nm bei Kilometer 50 entfällt in diesem Fall komplett, da die für diese Gewichtung optimale Steuerung zwischen den oben bereits erwähnten Stellen mit geringer Maximalgeschwindigkeit bei Kilometer 48 und Kilometer 51 lediglich auf 12 $\frac{\mathsf{m}}{\mathsf{s}}$ beschleunigt.

Dieses Verhalten der Geschwindigkeitstrajektorie ist über den gesamten Horizont zu beobachten. Aufgrund der sehr hohen Gewichtung der Momentänderungen wird das Einhalten der Wunschgeschwindigkeit nebensächlich. Daher ist an sehr vielen Stellen eine stark schwankende Geschwindigkeit – auch weit unter Wunschgeschwindigkeit – zu sehen. Gerade an steilen Anstiegen wie etwa bei Kilometer 38 oder 42 wird mit einem konstanten – wenn auch teilweise hohen – Motormoment gefahren und dadurch ein Verlust an Geschwindigkeit hingenommen.

Spätestens bei Betrachtung der Gangwahl wird deutlich, dass die Wahl der Gewichte für eine reale Fahrt eines LKW wahrscheinlich ungeeignet erscheint. Es existieren sehr wenige Phasen, in denen die Wahl des Gangs über eine Strecke von mehreren Kilometern beibehalten wird, der Plot ist vielmehr von sehr vielen Gangwechseln geprägt.

Diese beiden Beispiele für die Wahl der Gewichte sollen demonstrieren, welche Auswirkungen die Veränderungen der Gewichte haben können. Während das erste Beispiel[29] eine wirkliche Alternative zur in dieser Arbeit als Standard benutzten Wahl darstellen kann, ist das zweite Beispiel[30] aufgrund der oben genannten Gründe ungeeigneter. Selbstverständlich ist die Anzahl der sinnvollen Kombinationen sehr groß und hängt auch von den jeweils zu fahrenden Strecken ab, so dass die beiden Beispiele lediglich als Inspiration für weitere Untersuchungen dienen können.

[29] Siehe Abbildung 7.27.
[30] Siehe Abbildung 7.28.

Kapitel 8

Fazit und Ausblick

Im Rahmen dieser Arbeit haben wir einen LKW modelliert und darauf basierend ein gemischt-ganzzahliges Optimalsteuerungsproblem hergeleitet. Daraufhin haben wir gezeigt, dass sich die Methode der *Dynamischen Programmierung* für die Lösung dieser Probleme eignet. Im Vergleich mit der alternativen *direkten Mehrzielmethode* zeigten sich leicht bessere Ergebnisse der *Dynamische Programmierung* bei wesentlich höherer Laufzeit. Gerade bei der Wahl der diskreten Gänge kann eine Überlegenheit der *Dynamischen Programmierung* im Umgang mit nichtkontinuierlichen Steuerungen beobachtet werden.

Auch haben wir die *Dynamische Programmierung* um Techniken der *Nichtlinearen Modellprädiktiven Regelung* erweitert und konnten so auf einer 40 km langen Teststrecke Vergleiche zwischen einer global optimalen Steuerung und Steuerungen, die sich bei verschieden langen Prädiktionshorizonten ergeben, anstellen. Dabei konnten wir zeigen, dass eine Prädiktionslänge von 1000 m – zumindest auf der Vergleichsstrecke – ausreicht, um einen Zielfunktionswert zu erreichen, der bis auf 0.1 % an das globale Optimum heranreicht. Für kürzere Prädiktionshorizonte ergaben sich erwartungsgemäß schlechtere Ergebnisse.

Aufgrund der hohen algorithmischen Komplexität scheint es jedoch nicht sinnvoll, prädiktive *Dynamische Programmierung* im Online-Kontext einzusetzen, da entweder das Modell vereinfacht oder die Diskretisierungen stark vergröbert werden müssten, um Antwortzeiten erzielen zu können, die einem fahrenden LKW gerecht werden.

Des Weiteren haben wir durch die vollzogenen Sensitivitätsanalysen zeigen können, welchen Einfluss u.a. die Diskretisierungsfeinheiten der eigentlich reellwertigen Größen wie Ort, Momente oder Geschwindigkeit auf das Ergebnis der *Dynamischen Programmierung* haben können. Hier stellte sich heraus, dass die von uns häufig verwendete Diskretisierung $\delta_v = 0.1\frac{\mathsf{m}}{\mathsf{s}}$, $\delta_s = 50$ m, $\delta_M = 100$ Nm als ein sinnvoller Kompromiss zwischen hoher Laufzeit und geringer Approximationsgüte angesehen werden kann. Einzig die Diskretisierung der Momente sollte in Zukunft eventuell kleiner gewählt werden, da man mit der Bestrafung der Momentänderungen in Verbindung mit hohen Abständen zwischen zwei Gitterpunkten der Diskretisierung die Freiheit der Steuerung etwas einschränkt.

Außerdem wäre es bei einer weitergehenden Analyse mit Sicherheit interessant, die

Auswirkungen fehlerhafter Messdaten auf die Steuerung des LKW zu untersuchen. Wie sehr gerät beispielsweise der LKW von der optimalen Geschwindigkeitstrajektorie ab, wenn das vorausliegende Teilstück nicht – wie gemessen – 3 % sondern 4 oder gar 5 % Steigung hat? Gelingt es dabei eventuell mit der *Nichtlinearen Modellprädiktiven Regelung* bei Eintreffen der korrigierten Informationen schnell genug eine neue Steuerung zu berechnen, die den LKW wieder an die zuvor berehnete bzw. an eine für die geänderten Daten optimale Trajektorie heranführt? Und welche Einbußen in der Zielfunktion haben diese Störungen zur Folge?

Diese Fragen stellen sich natürlich nicht nur für den Einsatz der *Dynamischen Programmierung*, sondern auch für andere Optimalsteuerungsmethoden wie die in dieser Arbeit angeführten *direkten Mehrzielmethode*.

Eventuell ist auch von Interesse, ob die für die *Dynamische Programmierung* untersuchten Diskretisierungen größere Auswirkungen haben, wenn man sie nicht nur für die globale Methode, sondern auch für *Dynamische Programmierung* mit *Gleitendem Horizont* untersucht.

Somit ergeben sich auf diesem speziellen Gebiet weitere Fragen, die die zukünftige Forschung aufgreifen könnte.

Auch Erweiterungen auf komplexere Modelle, die beispielsweise andere Fahrzeuge und Staudaten berücksichtigen, sind denkbar und lassen zusätzliche Erkenntnisse erwarten. Davon abgesehen könnte die Fahrzeugdynamik beispielsweise auf Hybridfahrzeuge erweitert werden, da gerade die Kopplung des Energieverbrauchs in Verbindung mit optimaler Steuerung eine Möglichkeit darstellt, den Kraftstoffverbrauch weiter zu minimieren.

Letztendlich bleibt es spannend abzuwarten, ob die *Dynamische Programmierung* durch algorithmische Verbesserungen – und nicht etwa durch Vereinfachung der Modelle – in dem Maße beschleunigt werden kann, dass für die Länge des Prädiktionshorizonts global optimale Steuerungen online berechnet werden können, oder ob direkte Methoden zunehmend besser mit diskreten Steuerungen umgehen können werden und ob diese Techniken Einzug in die Fahrzeuge des Massenmarkts erhalten werden.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] J. Albersmeyer. Effiziente Ableitungserzeugung in einem adaptiven BDF-Verfahren. Diploma thesis, Universität Heidelberg, 2005.

[2] M. Back. *Prädiktive Antriebsregelung zum energieoptimalen Betrieb von Hybridfahrzeugen*. PhD thesis, Universität Karlsruhe, 2006.

[3] R. Bellman. *Dynamic Programming*. University Press, Princeton, 1957.

[4] D. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1 and 2. Athena Scientific, Belmont, MA, 1995.

[5] J. Betts. *Practical Methods for Optimal Control Using Nonlinear Programming*. SIAM, Philadelphia, 2001.

[6] T. Binder, L. Blank, H. Bock, R. Bulirsch, W. Dahmen, M. Diehl, T. Kronseder, W. Marquardt, J. Schlöder, and O. Stryk. Introduction to model based optimization of chemical processes on moving horizons. In M. Grötschel, S. Krumke, and J. Rambau, editors, *Online Optimization of Large Scale Systems: State of the Art*, pages 295–340. Springer, 2001.

[7] H. Bock and K. Plitt. A Multiple Shooting algorithm for direct solution of optimal control problems. In *Proceedings 9th IFAC World Congress Budapest*, pages 243–247. Pergamon Press, 1984. Available at http://www.iwr.uni-heidelberg.de/groups/agbock/FILES/Bock1984.pdf.

[8] M. Diehl. A direct multiple shooting method for the optimization and control of chemical processes. Master's thesis, Universität Heidelberg, 1998.

[9] M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*. PhD thesis, Universität Heidelberg, 2001. http://www.ub.uni-heidelberg.de/archiv/1659/.

[10] M. Diehl, H. Bock, and J. Schlöder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43(5):1714–1736, 2005.

[11] M. Diehl, H. Bock, J. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *J. Proc. Contr.*, 12(4):577–585, 2002.

[12] H. Ferreau. `qpOASES` – an open-source implementation of the online active set strategy for fast model predictive control. In *Proceedings of the Workshop on Nonlinear Model Based Control – Software and Applications, Loughborough*, 2007.

[13] R. Fletcher. Resolving degeneracy in quadratic programming. Numerical Analysis Report NA/135, University of Dundee, Dundee, Scotland, 1991.

[14] M. Gerdts. A variable time transformation method for mixed-integer optimal control problems. *Optimal Control Applications and Methods*, 27(3):169–182, 2006.

[15] E. Gertz and S. Wright. Object-oriented software for quadratic programming. *ACM Transactions on Mathematical Software*, 29:58–81, 2003.

[16] P. Gill, W. Murray, and M. Saunders. *User's Guide For QPOPT 1.0: A Fortran Package For Quadratic Programming*, 1995. Available at http://www.sbsi-sol-optimize.com/manuals/QPOPT%20Manual.pdf.

[17] L. Grüne, S. Sager, F. Allgöwer, H. Bock, and M. Diehl. *Produktionsfaktor Mathematik*, chapter Vorausschauend planen, gezielt handeln – über die Regelung und Steuerung technischer Prozesse, pages 27–62. acatech, 2008. ISSN 1861-9924, ISBN 978-3-540-89434-6.

[18] S. Han. Superlinearly convergent variable-metric algorithms for general nonlinear programming problems. *Mathematical Programming*, 11:263–282, 1976.

[19] E. Hellström. Explicit use of road topography for model predictive cruise control in heavy trucks. Master's thesis, Linköping University, SE-581 83 Linköping, 2005.

[20] E. Hellström. Look-ahead control of heavy trucks utilizing road topography. Master's thesis, Linköping University, 2007. LiU-TEK-LIC-2007:28, Thesis No. 1319.

[21] E. Hellström, M. Ivarsson, J. Åslund, and L. Nielsen. Look-ahead control for heavy trucks to minimize trip time and fuel consumption. *Control Engineering Practice*, 17:245–254, 2009.

[22] C. Kirches. A numerical method for nonlinear robust optimal control with implicit discontinuities and an application to powertrain oscillations. Diploma thesis, Universität Heidelberg, October 2006.

[23] C. Kirches, H. Bock, J. Schlöder, and S. Sager. Block structured quadratic programming for the direct multiple shooting method for optimal control. *Optimization Methods and Software*, 2010. (in review). Available Online: http://www.optimization-online.org/DB_HTML/2009/09/2395.html.

[24] C. Kirches, H. Bock, J. Schlöder, and S. Sager. Complementary condensing for the direct multiple shooting method. In H. Bock, E. Kostina, H. Phu, and R. Rannacher, editors, *Proceedings of the Fourth International Conference on High Performance Scientific Computing: Modeling, Simulation, and Optimization of Complex Processes, Hanoi, Vietnam, March 2–6, 2009*, Berlin Heidelberg New York, 2010. Springer Verlag. submitted.

[25] C. Kirches, S. Sager, H. Bock, and J. Schlöder. Time-optimal control of automobile test drives with gear shifts. *Optimal Control Applications and Methods*, 2010. DOI 10.1002/oca.892.

[26] D. Leineweber. Analyse und Restrukturierung eines Verfahrens zur direkten Lösung von Optimal-Steuerungsproblemen. Master's thesis, Universität Heidelberg, 1995.

[27] D. Leineweber, I. Bauer, H. Bock, and J. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part I: Theoretical aspects. *Computers and Chemical Engineering*, 27:157–166, 2003.

[28] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Verlag, Berlin Heidelberg New York, 1st edition, 1999. ISBN 0-387-98793-2 (hardcover).

[29] A. Potschka, H. Bock, and J. Schlöder. A minima tracking variant of semi-infinite programming for the treatment of path constraints within direct solution of optimal control problems. *Optimization Methods and Software*, 24(2):237–252, 2009.

[30] M. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In G. e. Watson, editor, *Numerical Analysis Dundee*, volume 3 of *Springer Verlag*, pages 144–157, Berlin, 1977.

[31] S. Sager. *Numerical methods for mixed–integer optimal control problems*. Der andere Verlag, Tönning, Lübeck, Marburg, 2005. ISBN 3-89959-416-9. Available at http://sager1.de/sebastian/downloads/Sager2005.pdf.

[32] S. Sager. Reformulations and algorithms for the optimization of switching decisions in nonlinear optimal control. *Journal of Process Control*, 19(8):1238–1247, 2009.

[33] S. Sager, H. Bock, and M. Diehl. The integer approximation error in mixed-integer optimal control. *Optimization Online*, 2:1–16, 2009. Submitted to Mathematical Programming A.

[34] S. Sager, H. Bock, M. Diehl, G. Reinelt, and J. Schlöder. Numerical methods for optimal control with binary control functions applied to a Lotka-Volterra type fishing problem. In A. Seeger, editor, *Recent Advances in Optimization (Proceedings of the 12th French-German-Spanish Conference on Optimization)*, volume 563 of

Lectures Notes in Economics and Mathematical Systems, pages 269–289, Heidelberg, 2006. Springer.

[35] S. Sager, C. Kirches, and H. Bock. Fast solution of periodic optimal control problems in automobile test-driving with gear shifts. In *Proceedings of the 47th IEEE Conference on Decision and Control (CDC 2008), Cancun, Mexico*, pages 1563–1568, 2008. ISBN: 978-1-4244-3124-3.

[36] S. Sager, G. Reinelt, and H. Bock. Direct methods with maximal lower bound for mixed-integer optimal control problems. *Mathematical Programming*, 118(1):109–149, 2009.

[37] S. Terwen, M. Back, and V. Krebs. Predictive powertrain control for heavy duty trucks. In *Proceedings of IFAC Symposium in Advances in Automotive Control*, pages 451–457, Salerno, Italy, 2004.

[38] R. Vanderbei. LOQO: An interior point code for quadratic programming. *Optimization Methods and Software*, 11(1–4):451–484, 1999.

Index

Erklärung

Hiermit versichere ich, dass ich meine Arbeit selbständig unter Anleitung verfasst habe, dass ich keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, und dass ich alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entlehnt sind, durch die Angabe der Quellen als Entlehnungen kenntlich gemacht habe.

Heidelberg, den 13. Januar 2010.