Numerische Simulation von Zugfahrten unter Realbedingungen

Diplomarbeit

Technische Universität Chemnitz
Fakultät für Mathematik

Eingereicht von: **Roman Unger**
geb. am: 16. Januar 1973
in: Stollberg

Betreuer: **Prof. Dr. Volker Mehrmann**
Dr. Wilfried Weinelt

8. März 2001

Aufgabenstellung

Entwicklung von numerischen Methoden zur Simulation von Zugfahrten unter der Berücksichtigung von Sicherheitsabständen, Sicherheitshaltebedingungen, Geschwindigkeitsbeschränkungen, Beschleunigungsbeschränkungen und Streckendaten.
Implementierung als Simulationswerkzeug zum Einsatz in realen Dispositionstools, zum Beispiel bei der Deutschen Bahn.

Vorwort

In den Betriebszentralen der Deutschen Bahn ist es für dispositive Zwecke nötig, den Laufweg eines bestimmten Zuges vorauszuberechnen, um unter anderem Aussagen zum Zeitverlauf der Fahrt zu bekommen.

Desweiteren ist eine Simulation des Ablaufs der Fahrten mehrerer Züge und ihrer gegenseitigen Beeinflussung auf dem Schienennetz zum einen hilfreich bei der langfristigen Konzipierung von Fahrplänen aber zum anderen auch ein wichtiges Werkzeug bei Dispositionsentscheidungen in Echtzeit für Umleitungen von Zügen oder anderem.

In den folgenden Kapiteln soll dazu die allgemeine Betrachtung dieser Problemklasse sowie der Zugang zur Simulation einzelner Züge und von Schienennetzen angeschnitten werden. Neben der theoretischen Betrachtung erfolgt die Entwicklung und Erprobung von Simulationsprogrammen in C++ zu diesen beiden Problemen.

An dieser Stelle möchte ich meinem Betreuer Prof. Dr. Volker Mehrmann (TU Berlin), Dr. Wilfried Weinelt (TU Chemnitz) sowie Dr. Christoph Blendinger (TLC GmbH, Frankfurt am Main) und Andreas Steinbrecher (TU Berlin) für die Unterstützung bei der vorliegenden Arbeit danken.

Inhaltsverzeichnis

Liste der verwendeten Bezeichnungen

A, B, B^i	Die Matrizen im Differentialgleichungssystem
$\beta_j \in \mathbf{N}_0$	Anzahlen von Sperrungen im Sicherungsblock j
$\chi_j(x^1 \dots x^{n_z})$	Indikatorfunktion zur Freischaltung von Sicherungsblock j
$\sigma^i(n, s)$	Zuordnungsfunktion für lokale und globale Numerierung der Sicherungsblöcke
$a^i_{min}(s)$	Zulässige Bremsverzögerung des i-ten Zuges
$b_j(s)$	Bremskurve im j-ten Sicherungsblock
$C(x, u)$	Eine allgemeine Steuerungsbeschränkung
$\texttt{entsperr}(j)$	PMA der Entsperrungen anderer Sicherungsblöcke beim Verlassen vom Sicherungsblock j
$f^0(t, x, u)$	Eine Bewertungsfunktion
$H(x, u, \lambda)$	Die Hamiltonfunktion
$J(\cdot)$	Ein Kostenfunktional
$L^i \subset \mathbf{R}$	Laufweg des i-ten Zuges
M	Die Steuerbarkeitsmatrix
$m_s \in \mathbf{N}$	Anzahl der Sicherungsblöcke
$n_z \in \mathbf{N}$	Anzahl der Züge
$\mathbf{N}$	Die natürlichen Zahlen
$N(x(t_i), t_i)$	Eine allgemeine Innere-Punkt-Bedingung
m	Masse des Zuges im Falle der Betrachtung eines Zuges
m^i	Masse des i-ten Zuges im Falle der Betrachtung mehrerer Züge
$\omega^i(s)$	Neigung der Strecke des i-ten Zuges
$\mathbf{R}$	Die reellen Zahlen
$\mathbf{R}^+$	Die positiven reellen Zahlen
$\mathbf{R}^-$	Die negativen reellen Zahlen
$\mathbf{R}^-_0$	$\mathbf{R}^- \bigcup \{0\}$
$\mathbf{R}^+_0$	$\mathbf{R}^+ \bigcup \{0\}$
$s(t)$	Eine zeitabhängige Ortsfunktion
s^i_j	Streckenpunkt Nummer j des i-ten Zuges
$S_j \subset L^i$	Sicherungsblock Nummer j
$S(x)$	Eine allgemeine Zustandsbeschränkung
$\texttt{sperr}(j)$	PMA der Sperrungen anderer Sicherungsblöcke bei Einfahren in den Sicherungsblock j

$u(t)$	Eine zeitabhängige Steuerungsfunktion
$v(s)$	Eine ortsabhängige Geschwindigkeitsfunktion
$v(t)$	Eine zeitabhängige Geschwindigkeitsfunktion
$v^i_{max}(s)$	Zulässige Höchstgeschwindigkeit des i-ten Zuges
$v^i_{max1}(t,s)$	Durch lokalen Sicherungsblock 1 bewirkte Geschwindigkeitsrestriktion des Zuges i
$v^i_{max2}(t,s)$	Durch lokalen Sicherungsblock 2 bewirkte Geschwindigkeitsrestriktion des Zuges i
$v^i_{rel}(t,s)$	Relevante Geschwindigkeitsrestriktion des Zuges i
$x^i(t)$	Vektor mit Ort und Geschwindigkeit des i-ten Zuges
$x^i_1(t)$	Ort des i-ten Zuges
$x^i_2(t)$	Geschwindigkeit des i-ten Zuges

Kapitel 1

Problem der einzelnen Zugfahrt

1.1 Allgemeine Formulierung als Optimalsteuerproblem

Wir betrachten die folgende einfache Strecke, der Startpunkt sei s^0 der Endpunkt s^n

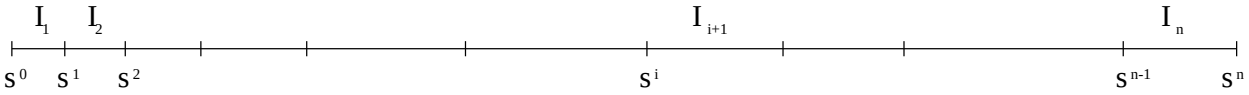

Abbildung 1.1: Einteilung der Strecke in Intervalle

Der Zustand des Systems wird durch die zum Zeitpunkt t erreichte Position $s(t)$ und die Geschwindigkeit $v(t)$ des Zuges beschrieben.

$$\begin{bmatrix} s(t) \\ v(t) \end{bmatrix} = x(t). \tag{1.1}$$

Dabei gelte die Anfangsbedingung

$$\begin{bmatrix} s(t_0) \\ v(t_0) \end{bmatrix} = x(t_0) = \begin{bmatrix} s^0 \\ v^0 \end{bmatrix} \tag{1.2}$$

und die folgende Bedingung, daß der Zug nach einer (endlichen) Zeit t_n im Zielpunkt s^n hält

$$\begin{bmatrix} s(t_n) \\ v(t_n) \end{bmatrix} = x(t_n) = \begin{bmatrix} s^n \\ v^n \end{bmatrix} \tag{1.3}$$

Gesteuert wird der Zug durch eine auf ihn einwirkende Kraft $u = u(t)$.
Aus der Newtonschen Bewegungs-Differentialgleichung $F = m\ddot{x}$ ergibt sich somit

$$m\ddot{s}(t) = u(t). \tag{1.4}$$

Dabei ist $u(t)$ die auf den Zug wirkende Gesamtkraft, die sich aus folgenden Anteilen zusammensetzt:

- Streckenabhängiger Anteil $u_f(s, v)$, der vom Triebfahrzeugsführer nicht beeinflußt werden kann. (Reibungskräfte durch Luftwiderstand oder Rollreibung, Hangabtriebskraft bei Steigung oder Gefälle)

- Die beeinflußbare Antriebs- bzw. Bremskraft des Zuges $u_v(t)$, die unsere Steuerfunktion darstellt und nach unten durch u_{min} (maximale Bremskraft des Zuges) sowie nach oben durch die Maximalleistung P_{max} des Triebfahrzeuges beschränkt ist.
  Mit $P = W/t$ und $W = Fs$ ergibt sich $P = Fv$, d.h. die maximal mögliche Antriebskraft ist nicht konstant, sondern geschwindigkeitsabhängig.

  Bei konstanter Maximalleistung ergibt sich somit für den Verlauf der maximalen Antriebskraft eine Funktion der Art Konstante / v .

  Im weiteren sei also eine monoton fallende Funktion $u_{max}(v)$ diese obere Schranke der Antriebskraft.

Es ergibt sich

$$u = u_f(s, v) + u_v(t) \qquad u_v(t) \in [u_{min}, u_{max}(v(t))] \tag{1.5}$$

Wir hatten

$$\begin{aligned} x(t) &= \begin{bmatrix} s(t) \\ v(t) \end{bmatrix} \\ \dot{s}(t) &= v(t) \\ \dot{v}(t) &= a(t) = \ddot{s}(t) = \frac{1}{m} u(t) \end{aligned}$$

und somit in Matrixnotation

$$\begin{aligned} \begin{bmatrix} \dot{s}(t) \\ \dot{v}(t) \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} s(t) \\ v(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u(t) \\ \dot{x}(t) &= Ax(t) + Bu(t) \end{aligned}$$

Das Problem der zeitoptimalen Steuerung ohne Streckenparameter ist dann

$$\begin{cases} (t_n - t_0) \rightarrow \min \\ \dot{x}(t) = Ax(t) + B[u_f(s, v) + u_v(t)] \\ x(t_0) = \begin{bmatrix} s^0 \\ v^0 \end{bmatrix} \\ x(t_n) = \begin{bmatrix} s^n \\ v^n \end{bmatrix} \\ u_v(t) \in [u_{min}, u_{max}(v(t))]. \end{cases} \tag{1.6}$$

Weiterhin sind aber noch folgende Zustandsbeschränkungen, die durch vorgegebene Streckenparameter festgelegt sind, zu erfüllen:

- zulässige Höchstgeschwindigkeit des Zuges in einem Abschnitt und Verbot des Rückwärtsfahrens:
  $v((s(t)) \in [0, v^i_{max}]$ für $s \in [s^{i-1}, s^i] \quad i = 1 \ldots n$

  In praktischen Situationen ist es sogar möglich, daß der Zug eine bestimmte Minimalgeschwindigkeit nicht unterschreiten darf. Ein Beispiel dafür ist ein schwerer Güterzug an einer starken Steigung. In diesen Fällen ist als untere Schranke eine nichtnegative Funktion $v_{min}(s(t))$ einzuhalten.

- Haltepunkte bzw. Ankunfts- und Abfahrtszeiten, d.h. der Punkt s^i ist im Zeitfenster $t^i_a \pm \epsilon^i$ zu erreichen und im Zeitfenster $t^i_s \pm \epsilon^i$ zu verlassen. Dabei stellen die ϵ^i einen Parameter dar, der die zulässige Abweichung vom Fahrplan in dem jeweiligen Streckenabschnitt angibt.

1.2 Betrachtung in der st-Ebene

Betrachtet man nun dieses Problem in der st-Ebene wie in Abbildung 1.2 ergibt sich folgendes:

Gesucht ist eine Steuerung u(t), die einen Weg-Zeit-Verlauf der Bewegung des Zuges bewirkt, der sich als Funktion s=s(t) darstellen lässt, die den Punkt $\left[\begin{smallmatrix} t_0 \\ s^0 \end{smallmatrix}\right]$ mit dem Punkt $\left[\begin{smallmatrix} t_n \\ s^n \end{smallmatrix}\right]$ verbindet. Dabei ist t_n nicht von vornherein festgelegt, sondern es soll gelten, daß t_n minimal werden soll.
Die vorhin angesprochenen Restriktionen gehen über zu:

- s(t) ist monoton nichtfallend, da stets $v(t) \geq 0$ gelten soll.

- Der Anstieg $\dot{s}(t) = v(t)$ ist nicht größer als die zulässige Höchstgeschwindigkeit in diesem Bereich.

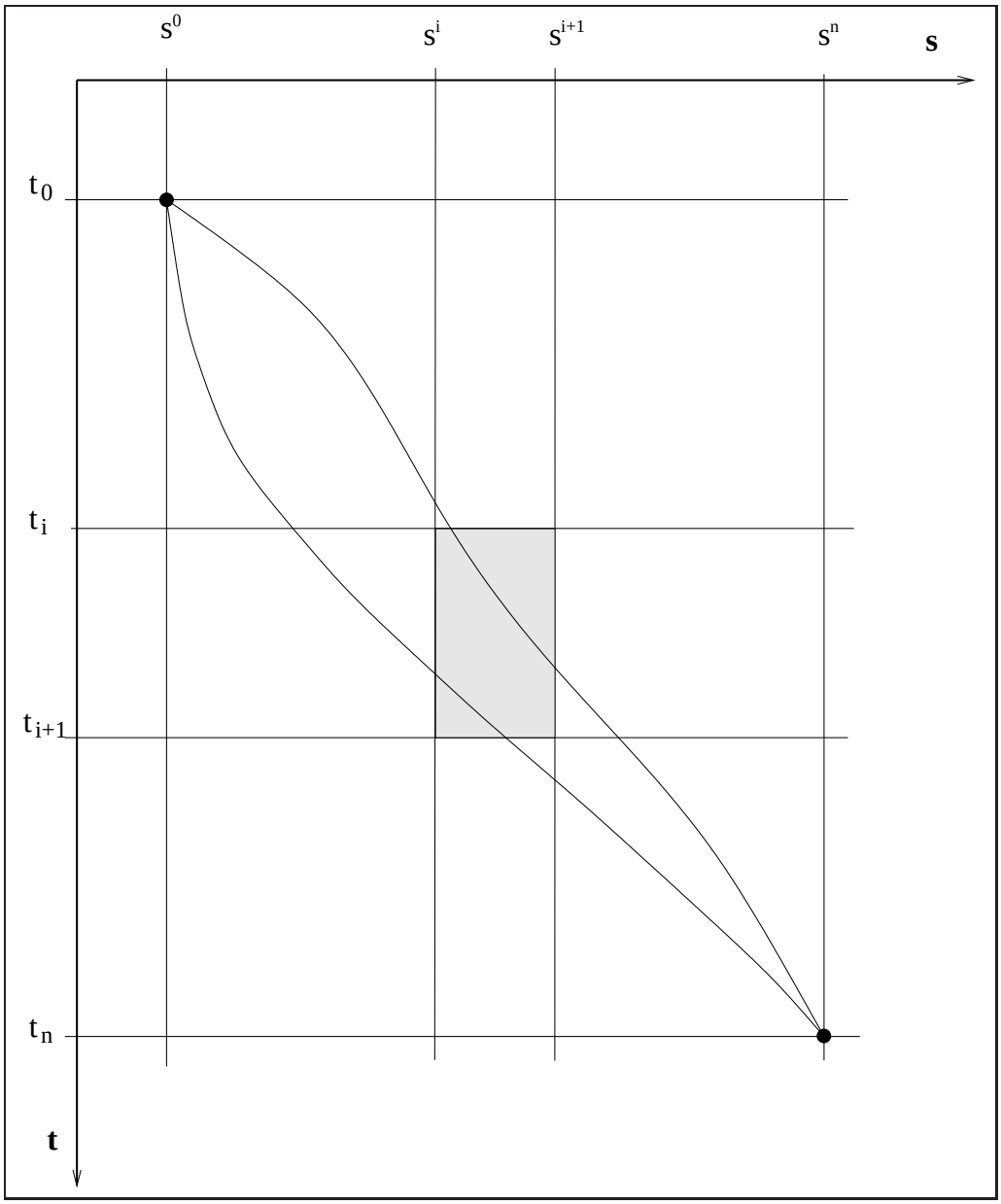

Abbildung 1.2: Darstellung in der st-Ebene

Kapitel 2

Das zeitoptimale Problem

Für die zeitliche Optimierung sind die Halte- und Abfahrtszeiten in einzelnen Haltepunkten nicht relevant, sie würden nur einen additiven Offset zur Optimalzeit darstellen.
Also gehen in die Aufgabe (1.6) nur die Geschwindigkeitsbeschränkungen ein.
Wir erhalten somit als Aufgabe zur zeitoptimalen Steuerung

$$\begin{cases} (t_n - t_0) \rightarrow \min \\ \dot{x}(t) = Ax(t) + B[u_f(s,v) + u_v(t)] \\ x(t_0) = \begin{bmatrix} s^0 \\ v^0 \end{bmatrix} \\ x(t_n) = \begin{bmatrix} s^n \\ v^n \end{bmatrix} \\ v(s(t)) \in [v_{min}(s(t)), v_{max}(s(t))] \\ u_v(t) \in [u_{min}, u_{max}(v(t))]. \end{cases} \tag{2.1}$$

Definition 1 (Steuerbarkeitsmatrix) *Die Steuerbarkeitsmatrix M ist definiert als folgende Blockmatrix:*

$$M = \left[B, AB, A^2B, \ldots, A^{n-1}B\right]. \tag{2.2}$$

Es handelt sich bei Aufgabe (2.1) um ein lineares autonomes System (A, B sind nicht zeitabhängig). Betrachtet man die Aufgabe zunächst ohne die Zustandsbeschränkungen, dann besagt die Kalman-Bedingung [1], daß derartige Systeme stabil steuerbar sind, wenn die Steuerbarkeitsmatrix M vollen Rang n hat.
Wir haben den Fall n=2, also:

$$\begin{aligned} M &= [B, AB] && (2.3) \\ &= \left[\begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}\right] && (2.4) \\ &= \begin{bmatrix} 0 & \frac{1}{m} \\ \frac{1}{m} & 0 \end{bmatrix}. && (2.5) \end{aligned}$$

Damit ist $rank(M) = n = 2$ und das System ist stabil steuerbar.
Der folgende Satz aus [1] liefert im Falle einer unbeschränkten Steuerung sogar die vollständige Steuerbarkeit, d.h. jeder Ausgangspunkt kann in endlicher Zeit in jeden Zielpunkt überführt werden.

Satz 1 *Sei $rank(M) = n$ und es gelte $Re(\lambda) \leq 0$ für alle Eigenwerte von A. Dann ist das System vollständig steuerbar.*

Wir haben als einzigen Eigenwert von A den Wert $\lambda = 0$, also $Re(\lambda) \leq 0$ ist erfüllt.
Das Problem in unserem Fall ist aber die Zustandsbeschränkung, wodurch die einfachen Methoden der optimalen Steuerung nicht anwendbar sind. Deshalb wird zuerst eine numerische Simulation durchgeführt.

2.1 Eine einfache numerische Simulation des zeitoptimalen Problems

Um einen ersten Einblick zu bekommen, simulieren wir eine Fahrt mit maximal zulässiger und möglicher Geschwindigkeit.
Wir verwenden konstante Zeitschrittweiten und bestimmen die Bremswege immer neu.
Es wird sich zeigen, daß dieses Herangehen zwar einfach, aber nicht sehr effektiv ist.
Zuerst ein paar Begriffe:

Definition 2 (Zulässigkeitskegel KZ(t)) *Der Zulässigkeitskegel KZ(t) sei die Menge aller Punkte s(t), die im Zeitintervall [0,t] angesteuert werden dürfen, ohne die maximal zulässige Geschwindigkeit zu überschreiten.*

Bemerkung 1 *Dabei wird nicht berücksichtigt, ob überhaupt eine Steuerung existiert, die diese Punkte erreichen kann.*

Definition 3 (Ansteuerbarer Kegel KS(t)) *Der ansteuerbare Kegel KS(t) sei die Menge aller Punkte s(t), die im Zeitintervall [0,t] mit maximal zulässiger Steuerung ohne Berücksichtigung von Geschwindigkeitsbeschränkungen angesteuert werden können.*

Wir geben eine Maximalzeit der Simulation (`maxzeit`) und die Anzahl der Diskretisierungspunkte (`ntdis`) vor.
Daraus ergibt sich die Zeitschrittweite Δt als $\Delta t =$ `maxzeit/ntdis`. Desweiteren seien gegeben:

- $n \in \mathrm{N}$ ... die Anzahl der Streckenintervalle, wobei der Anfangspunkt des letzten Intervalls der Zielpunkt s^n sei
- $s = [s_1 \dots s_n]^T \in \mathrm{R^n}$ ... die Endpunkte der einzelnen Streckenabschnitte (Startpunkt des ersten Streckenabschnitts sei $s_1 = 0$)
- $\mathtt{vmax} = [\mathtt{vmax_1} \dots \mathtt{vmax_n}] \in \mathrm{R^n_+}$ ... die zulässigen Maximalgeschwindigkeiten in den einzelnen Streckenabschnitten. Falls der Zug am Ende der Strecke halten soll, dann setzen wir $\mathtt{vmax_n} = 0$.

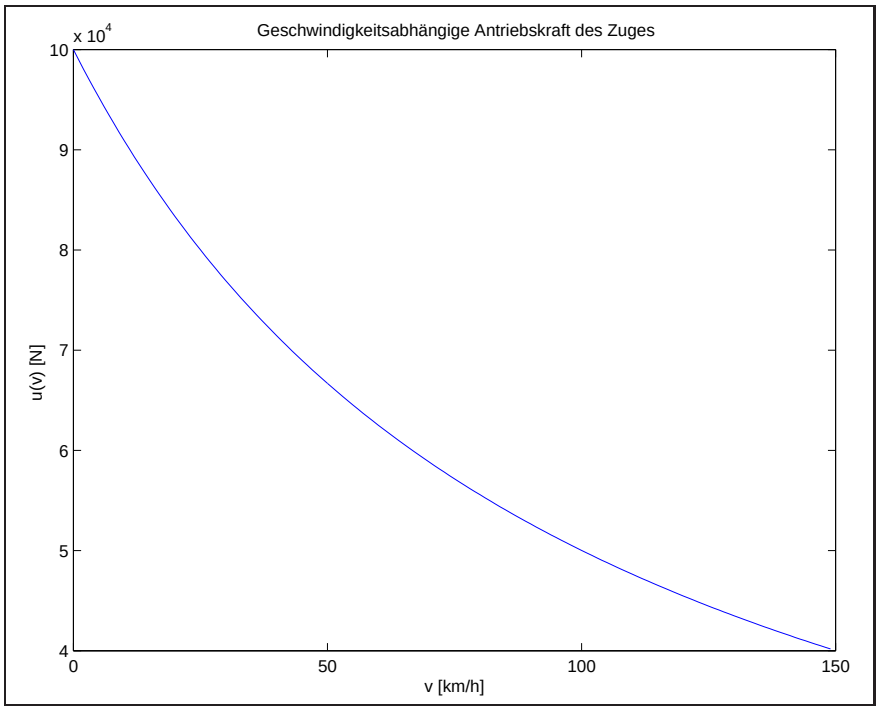

Abbildung 2.1: Geschwindigkeitsabhängige obere Schranke der Antriebskraft

- $\mathtt{a}_1, \mathtt{a}_2, \mathtt{a}_3 \in \mathrm{R}$ . . . Parameter zur Approximation der geschwindigkeitsabhängigen oberen Schranke des variablen Anteils der Antriebskraft des Triebfahrzeugs. (siehe Abbildung 2.1)

$$u_{max}(v) = \frac{\mathtt{a}_1}{\mathtt{a}_2 v + \mathtt{a}_3}$$

Im folgenden werden Zulässigkeitskegel, ansteuerbarer Kegel sowie eine Fahrt simuliert und mit Grafiken eines Beispiels auf einer Strecke von 9000 Metern dargestellt. Eine mögliche Variante der Geschwindigkeitsbeschränkungen in den Streckenabschnitten zeigt die Abbildung 2.2.
Davon ausgehend ist als erstes der Zulässigkeitskegel $KZ(t)$ in jedem Zeitschritt zu bestimmen. Dazu wird im i-ten Zeitpunkt festgestellt, in welchem Streckenabschnitt man sich befindet und abhängig davon die Maximalgeschwindigkeit bestimmt. Der $(i+1)$-te Ortspunkt $KZ(t_{i+1})$ ergibt sich dann als $KZ(t_{i+1}) := KZ(t_i) + \Delta t \cdot \mathtt{vmax(j)}$, wobei $KZ(t_i) \in (s_{j-1}, s_j]$. Startwert ist natürlich $KZ(t_0) = 0$. Dabei ergibt sich ein Verlauf wie in Abbildung 2.3.
Analog bestimmt man $KS(t)$ durch Simulation der Bewegung mit maximal zulässiger Steuerung, also $u_{max}(v)$ indem in jedem Diskretisierungsintervall die Antriebskraft als konstant angenommen wird, und der nächste Ortspunkt als Bewegung mit konstanter Beschleunigung ($s = a/2 \cdot t^2 + v_0 t + s_0$ wobei $a := \frac{1}{m} u_{max}(v_0)$) bestimmt wird. (Abbildung 2.4)
Die Idee der Simulation ist nun, die Strecke möglichst mit maximal zulässiger Geschwindigkeit abzufahren. Dazu wird in jedem Zeitpunkt der Bremsweg berechnet, den man benötigt um die Maximalgeschwindigkeit im folgenden Intervall einzuhalten. Dieser Brems-

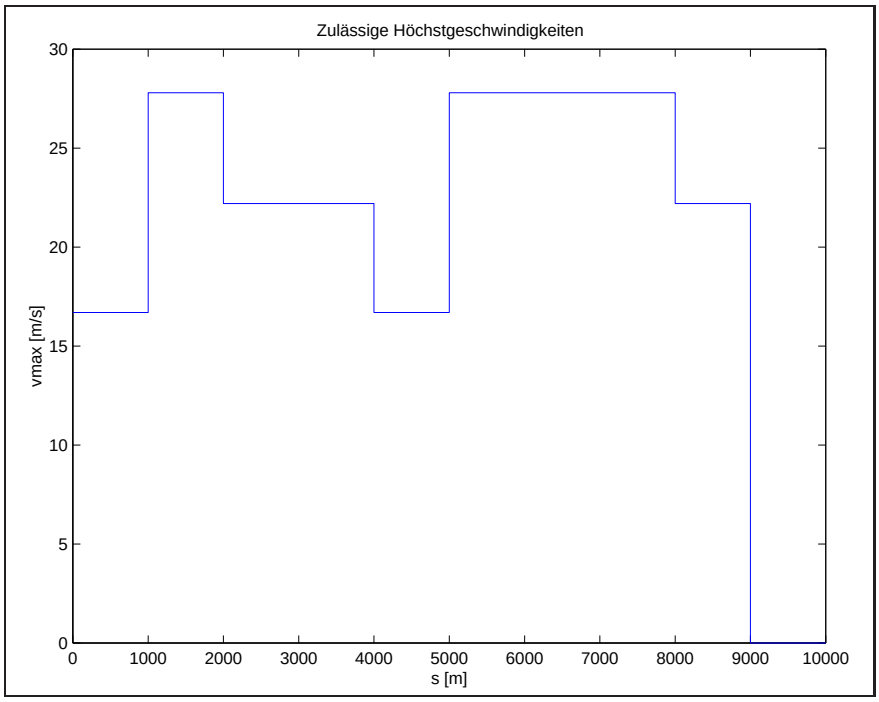

Abbildung 2.2: Zulässige Höchstgeschwindigkeiten auf den Streckenabschnitten

weg wird mit der zu Verfügung stehenden Reststrecke im Intervall verglichen und die Steuerung wird folgendermaßen geschaltet:

$$u(t_i) = \begin{cases} u_{min} & \text{falls} \quad Bremsweg \leq Restweg \\ \begin{cases} u_{max}(v(t_{i-1})) & \text{falls} \quad v(t_{i-1}) < vmax \\ 0 & \text{falls} \quad v(t_{i-1}) \geq vmax \end{cases} \end{cases} \tag{2.6}$$

Es ergibt sich ein Verlauf der Fahrt wie in den Abbildungen 2.5, 2.6 und 2.7.
Diese Simulation gewährt einen ersten Einblick in den möglichen Verlauf der zeitoptimalen Steuerung, für praktische Zwecke ist ein derartiges Vorgehen aber unbrauchbar, da sich zum einen durch das ständige Vorausberechnen der Bremswege zu große Laufzeiten ergeben und zum anderen das einfache Polygonzugverfahren zur Lösung der Differentialgleichungen durch Ungenauigkeiten zum „Rattern“ der Steuerung führt.
In (2.1) wurde die zeitoptimale Aufgabe als Optimalsteuerproblem mit Zustandsbeschränkung formuliert. Nun ist der nächste Schritt, eine Formulierung zu finden, die sich numerisch lösen lässt.
Dafür soll im weiteren erst einmal eine allgemeine Betrachtung dieser Problemklasse durchgeführt werden.

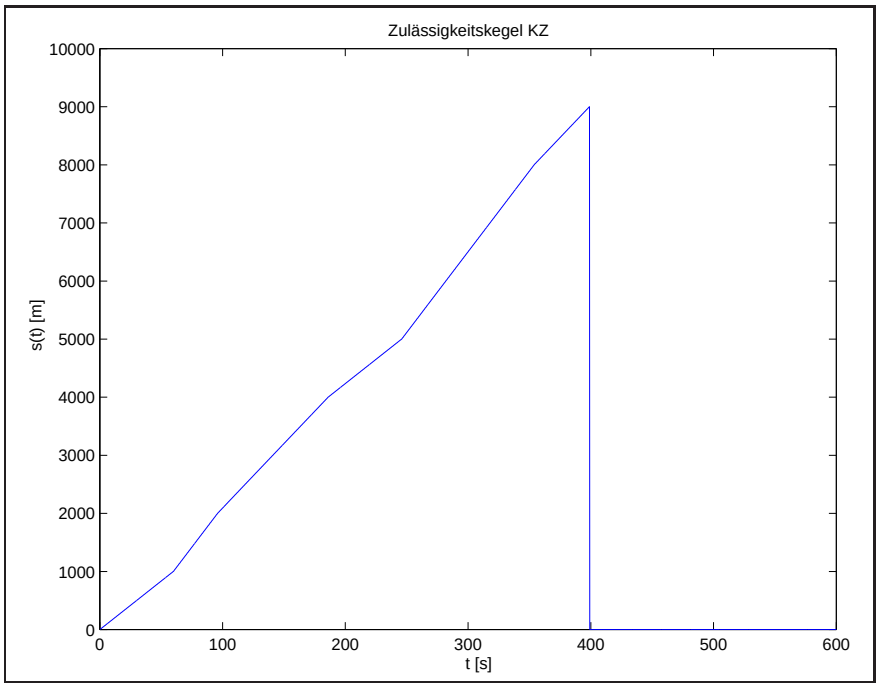

Abbildung 2.3: Der mit maximal zulässiger Geschwindigkeit ansteuerbare Kegel

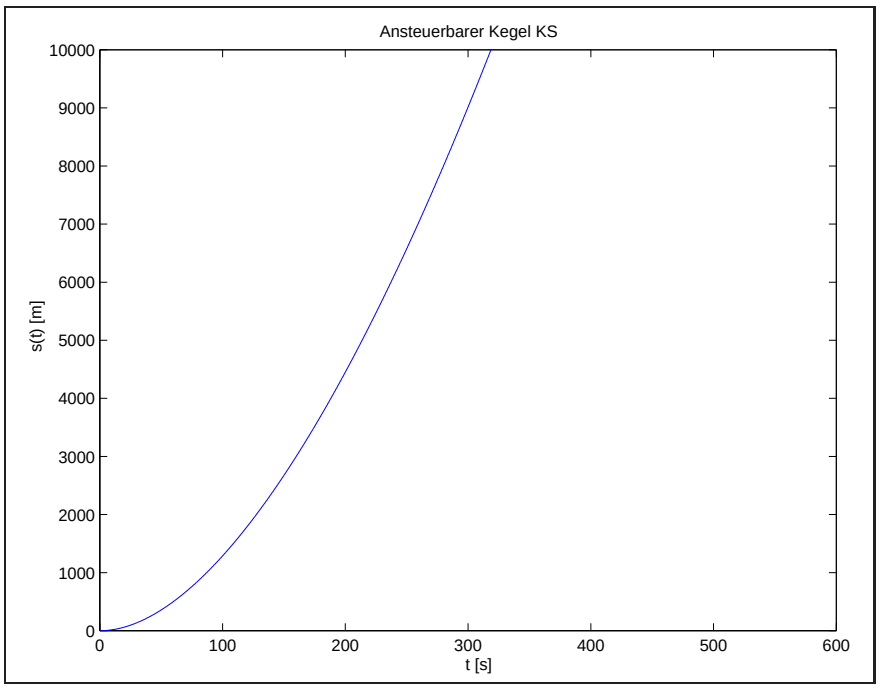

Abbildung 2.4: Der mit maximal zulässiger Steuerung ansteuerbare Kegel

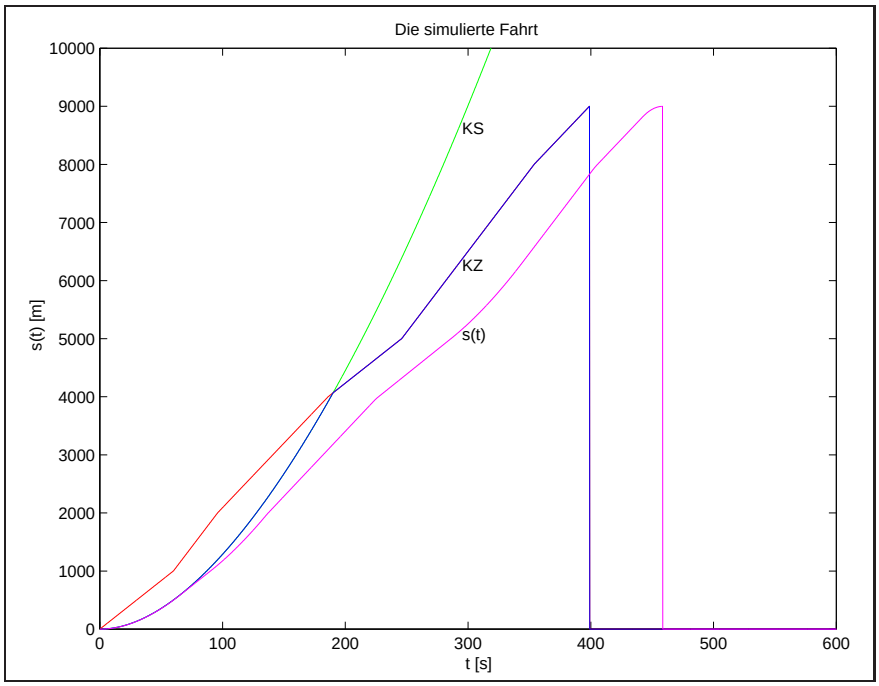

Abbildung 2.5: Die simulierte Fahrt mit KZ und KS

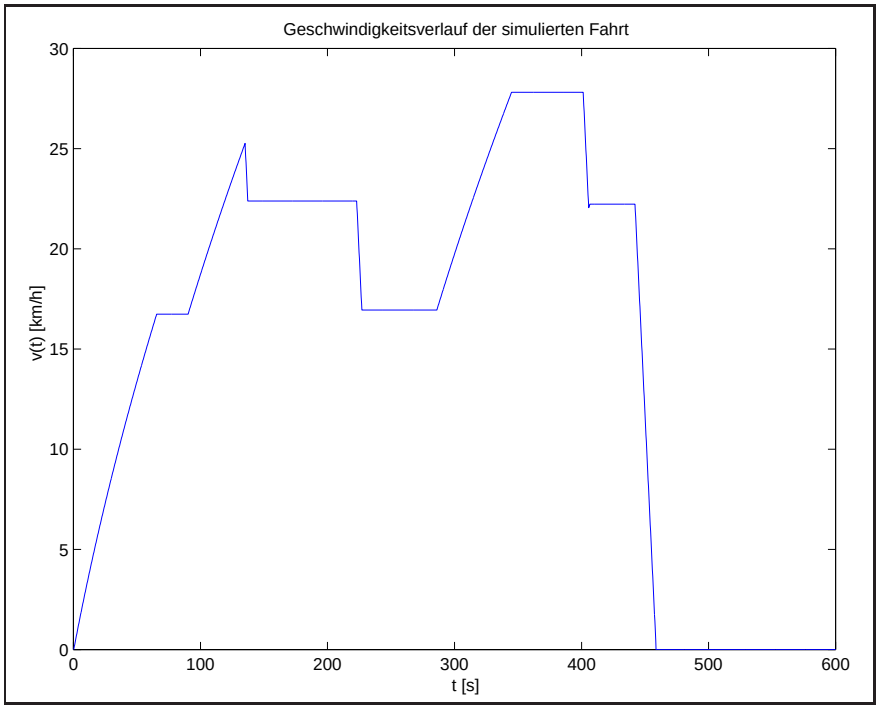

Abbildung 2.6: Der Geschwindigkeitsverlauf der simulierten Fahrt

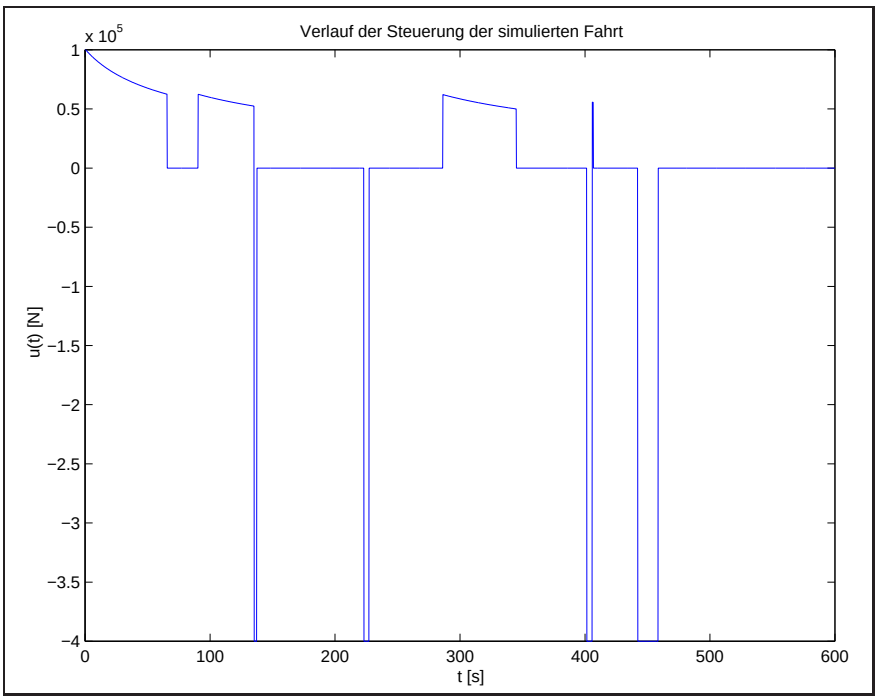

Abbildung 2.7: Verlauf der Steuerung während der simulierten Fahrt

Kapitel 3

Allgemeines zu Optimalsteuerproblemen mit Beschränkungen

Die Betrachtungen dieses Abschnitts basieren unter anderem auf Ausführungen in [2].

3.1 Überführung in ein Minimierungsproblem

Um Optimalsteuerprobleme lösen zu können, bietet sich die Überführung in ein äquivalentes Minimierungsproblem an.
Im Gegensatz zu klassischen Minimierungsaufgaben aus der Optimierungstheorie ist hier jedoch der Suchraum nicht endlichdimensional, sondern es handelt sich um unendlichdimensionale Funktionenräume.
Zunächst zwei neue Begriffe, die Bewertungsfunktion und das Kostenfunktional.

Definition 4 (Bewertungsfunktion und Kostenfunktional) *Eine vorgegebene Funktion*

$$f^0(t,x,u) : (0,\infty) \times \mathrm{R}^{\mathrm{n}} \times \mathrm{R}^{\mathrm{k}} \to \mathrm{R} \tag{3.1}$$

wird als Bewertungsfunktion bezeichnet. Das Funktional

$$J(u(.)) = \int_{t_0}^{t_1} f^0(t,x,u)dt \tag{3.2}$$

wird das, durch die Bewertungsfunktion f^0 erzeugte, Kostenfunktional genannt.

Damit wird die Aufgabe des Findens einer Optimalsteuerung zur Aufgabe, das Funktional J zu minimieren.

3.2 Zustand des Systems und Beschränkungen

Betrachten wir einmal ganz allgemein ein Optimalsteuerproblem. Man unterscheidet den Zustand des Systems

$$x : [t_0, t_f] \to \mathrm{R}^{\mathrm{n}}$$

und die Steuerung

$$u : [t_0, t_f] \to \mathrm{R}^{\mathrm{k}}$$

in dem Problem

$$\min \quad J = \phi(x(t_f), t_f) + \int_{t_0}^{t_f} f^0(t, x(t), u(t))dt \tag{3.3}$$

unter den Bedingungen

$$\dot{x} = f(x, u) \qquad f : \mathrm{R}^{\mathrm{n}} \times \mathrm{R}^{\mathrm{k}} \to \mathrm{R}^{\mathrm{n}} \tag{3.4}$$

$$x_i(t_0) = x_i^0 \in \mathrm{R} \quad \text{gegeben für } i = 1 \ldots n \tag{3.5}$$

$$\Psi(x(t_f), t_f) = 0 \quad \Psi : \mathrm{R}^{\mathrm{n}} \times \mathrm{R}_+ \to \mathrm{R}^{\mathrm{q}} \tag{3.6}$$

Dabei ist $t_f \in \mathrm{R}_+$ entweder fest oder frei. Im Falle fester Endzeit soll $q \leq n - 1$ gelten, anderenfalls $q \leq n$. q ist die Anzahl der durch die Funktion Ψ fest vorgegebenen Endbedingungen des Zustands.
Man sieht, daß bei fester Endzeit nicht alle Zustandsvariablen auch fest vorgegeben werden können. Der Term $\phi(x(t_f), t_f)$ im Zielfunktional J stellt hierbei eine Bewertung des Endzustandes dar. Es ist offensichtlich, daß nur eine Bewertung freier Endwerte sinnvoll ist, d.h. in ϕ sollten nur die $(n - q)$ Komponenten von x bewertet werden, die nicht in Ψ festgelegt wurden.

Weiterhin sind Steuerungsbeschränkungen

$$C(x, u) \leq 0 \qquad C : \mathrm{R}^{\mathrm{n}} \times \mathrm{R}^{\mathrm{k}} \to \mathrm{R}^{\mathrm{l}} \tag{3.7}$$

und Zustandsbeschränkungen

$$S(x) \leq 0 \qquad S : \mathrm{R}^{\mathrm{n}} \to \mathrm{R}^{\bar{\mathrm{l}}} \tag{3.8}$$

sowie Innere-Punkt-Bedingungen

$$N(x(t_I), t_I) = 0 \qquad N : \mathrm{R}^{\mathrm{n}} \times [0, t_f) \to \mathrm{R}^{\bar{\mathrm{q}}} \tag{3.9}$$

vorgegeben und wir setzen voraus, daß alle Funktionen hinreichend oft differenzierbar sind.

Definition 5 (Hamiltonfunktion) *[1] Mit obigen Bezeichnungen sei für ein Optimalsteuerproblem folgende Funktion*

$$H(x,u,\lambda) = f^0(t,x,u) + \lambda^T f(x,u) \tag{3.10}$$

$$\lambda : [0,t_f] \to \mathrm{R^n}$$

als Hamiltonfunktion H definiert.

Desweiteren sei die Hilfsfunktion Φ definiert als

$$\Phi(x,t,\nu) = \phi(x,t) + \nu^T \Psi(x,t) \qquad \nu \in \mathrm{R^q} \tag{3.11}$$

Diese Funktion verbindet fest vorgegebene Endwerte mit freien Endwerten, die im Zielfunktional bewertet werden.
Mit Hilfe der Hamiltonfunktion lassen sich die beiden notwendigen Bedingungen an eine optimale Steuerung $\bar{u}(t)$ formulieren zu

$$H_x(\bar{x},\bar{u},\bar{\lambda})(x-\bar{x}) \geq 0 \qquad \forall x(\cdot) \tag{3.12}$$
$$H_u(\bar{x},\bar{u},\bar{\lambda})(u-\bar{u}) \geq 0 \qquad \forall u(\cdot) \tag{3.13}$$

Dabei sei mit H_x die partielle Ableitung der Hamiltonfunktion H nach x und analog mit H_u die partielle Ableitung nach u bezeichnet.
Auf diesem Weg läßt sich das allgemeine Optimalsteuerproblem in ein Randwertproblem überführen.
Zur Lösung dieses Problems sind aber noch die Umschaltpunkte der Steuerung zu bestimmen. Diese bestimmen die rechte Seite der Differentialgleichung in den einzelnen Abschnitten.
In unserem Fall der zeitoptimalen Steuerung ist jedoch nach Bestimmung der Schaltpunkte auch gleich die optimale Steuerung mit bestimmt, da im zeitoptimalen Fall die Steuerung nur ihre Extremalwerte (bang-bang-Steuerung) bzw. bei Fahrt auf den Zustandsbeschränkungen den entsprechenden (bekannten) Wert zur Kompensation äußerer Kräfte annimmt.

Kapitel 4

Berechnung der Schaltpunkte des zeitoptimalen Problems

Die numerische Berechnung der Auf- und Absprungpunkte bzw. Umschaltpunkte in den einzelnen Wegintervallen erfolgt durch Bestimmung der Schnittpunkte der Trajektorien der Geschwindigkeitsbeschränkung $v_{max}(s(t))$ mit den Trajektorien von $v(t)$ bei Fahrt mit extremaler Steuerung.
Am einfachsten ist das Problem lösbar, wenn folgende Einschränkungen gelten:

4.1 Bedingungen an die Zustandsbeschränkungen

1. Restriktionen in einem Wegintervall dürfen sich nur auf die Steuerung im vorherigen Intervall auswirken, nicht jedoch auf noch weiter zurückliegende Intervalle. Diese Bedingung sichert, daß Absprungpunkte von den Startbedingungen im Folgeintervall aus rückwärts bestimmt werden können. Abbildung 4.1 stellt ein Beispiel dar, was bei Verletzung dieser Bedingung passiert.

   Der Anstieg der Geschwindigkeitsbeschränkung darf nur so groß sein, daß nach einem Aufsprung die Steuerung in der Lage ist, die Restriktion bis zu einem eventuellen Absprungpunkt, in dem auf "Bremsen "umgeschaltet wird, mitzufahren. Damit wird in dem Intervall die Struktur gesichert, daß maximal ein Aufsprungpunkt und maximal ein Absprungpunkt vorliegen darf. Siehe dazu auch Abbildung 4.2

2. Bei fallender Geschwindigkeitsrestriktion, ist zu sichern, daß die minimale Steuerung (=maximale Bremskraft) in der Lage ist, diesem Fallen zu folgen.

Im weiteren sei die Geschwindigkeitsrestriktion unter Beachtung dieser Bedingungen polynomial approximiert durch

$$v_{max}(s) = p_0 + p_1 s + p_2 s^2 + p_3 s^3 \tag{4.1}$$

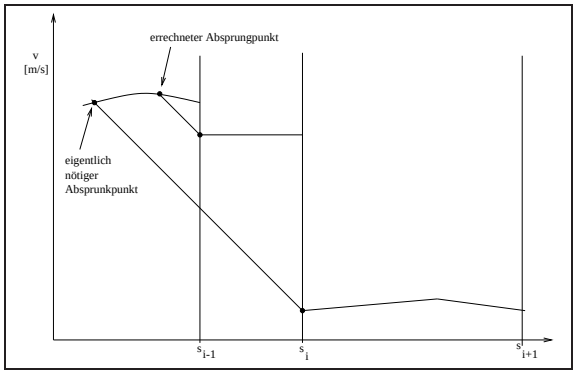

Abbildung 4.1: Erste Bedingung an die Geschwindigkeitsbeschränkung

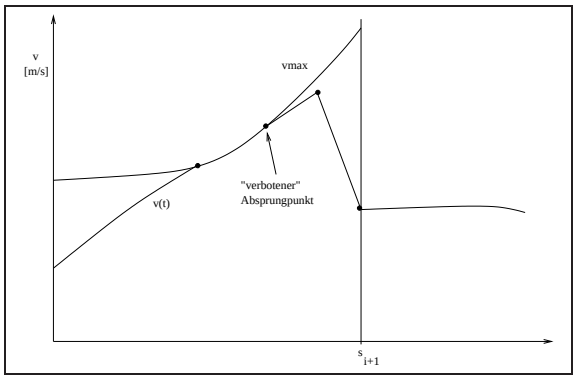

Abbildung 4.2: Zweite Bedingung an die Geschwindigkeitsbeschränkung

4.2 Bestimmung des Aufsprungpunktes

Wir betrachten im Punkt s^i den Fall, daß die gefahrene Geschwindigkeit in diesem Punkt kleiner ist als die Restriktion $v_{max}(s^i)$. Bekannt sind im Punkt s^i der zugehörige Zeitpunkt t_i, die momentane Geschwindigkeit $v(t_i)$ sowie die Steuerung $u_v = u_{max}(v(t_i))$. Gesucht ist der Zeitpunkt $\hat{t}$ in dem gilt: $\hat{v} := v(\hat{t}) = v_{max}(s(\hat{t}))$. Dabei soll gelten, daß $s(\hat{t}) \leq s^{i+1}$, denn sonst liegt in diesem Intervall kein Aufsprungpunkt vor. Dies ist bei der numerischen Umsetzung zu beachten. Wir definieren folgende, von t abhängige Funktion:

$$\sigma(t) := v_{max}(s(t)) - v(t) \tag{4.2}$$

Damit wird das Problem des Aufsprungpunktes zur Nullstellenbestimmung von (4.2). Dies erfolgt durch ein Bisektionsverfahren, unter anderem beschrieben in [10]:

1. Gegeben seien Genauigkeitsschranke ϵ, Startschrittweite Δt, die Startrichtung $r = +1$ sowie die Anfangswerte t_i, s^i und v^i

2. Setze $t = t_i + \Delta t$

3. Berechne $\sigma(t)$; bestimme dabei den Anteil $v(t)$ durch Lösung des Anfangswertproblems:
$$\begin{cases} \dot{x}(t) = Ax(t) + B[u_f(s,v) + u_{max}(v(t))] \\ x(t_i) = \begin{bmatrix} s^i \\ v^i \end{bmatrix} \end{cases} \tag{4.3}$$

4. Falls $|\sigma(t)| < \epsilon$ dann setze $\hat{t} := t$ und fertig.

5. Falls $\sigma(t) < 0$, dann ist die Geschwindigkeitsrestriktion verletzt. Wenn die Richtung r positiv ist, dann umkehren und Schrittweite halbieren, d.h. $r := -1$; $\Delta t := \frac{1}{2}\Delta t$.

6. Falls $\sigma(t) > 0$, dann ist die Geschwindigkeitsrestriktion noch nicht erreicht. Wenn die Richtung r negativ ist, dann umkehren und Schrittweite halbieren, d.h. $r := +1$; $\Delta t := \frac{1}{2}\Delta t$.

7. Gehe nun zurück zu Schritt 2.

Numerisch gibt es dabei noch folgendes zu beachten:

1. Falls in dem Intervall kein Aufsprungpunkt existiert, muss überwacht werden, ob bei der iterativen Bestimmung von $\hat{t}$ der Fall $\hat{s} > s^{i+1}$ eintritt, dann existiert kein Aufsprungpunkt in diesem Intervall, die Geschwindigkeit bleibt immer unterhalb der Restriktion, das Intervall wird mit maximaler Steuerung durchfahren.

2. Es ist möglich die Streckenparameter so zu wählen, daß die tatsächliche Steuerung $u = u_f + u_{max}$ zu Null wird (Extreme Steigung). In diesem Fall wird davon ausgegangen, daß die tatsächliche Steuerung u dann auf dem Wert Null bleibt. Praktisch bedeutet dies, daß der Zug am Berg „hängenbleibt“ , aber die Bremskraft des Zuges in jedem Fall stark genug ist, ein Zurückrollen zu verhindern. Für die Bestimmung der Schaltpunkte wird in diesem Fall das Verfahren abgebrochen, da eine Weiterfahrt nicht mehr möglich ist.

4.3 Bestimmung des Absprungpunktes

Die Bestimmung des Absprungpunktes ist nötig, wenn die Geschwindigkeitsrestriktion an der Intervallgrenze nach unten springt, d.h. $v_{max}(s^{i+1} + 0) < v_{max}(s^{i+1} - 0)$. Die Idee des Verfahrens ist, in dem Punkt s^{i+1} mit der dort vorliegenden rechtsseitigen zulässigen Maximalgeschwindigkeit zu starten, wobei mit negativer minimaler variabler Steuerung gefahren wird. Dabei wird zunächst als Startwert für t und s Null gewählt. Analog wie bei der Bestimmung des Aufsprungpunktes wird auch hier ein Bisektionsverfahren zur Nullstellenbestimmung von der Funktion $\sigma(t)$, definiert wie in (4.2) eingesetzt. Zu beachten ist dabei, daß man bei der Lösung des Anfangswertproblems

$$\begin{cases} \dot{x}(t) = Ax(t) + B[-u_f(s,v) - u_{min}(v(t))] \\ x(0) = \begin{bmatrix} 0 \\ v_{max(s^{i+1}+0)} \end{bmatrix} \end{cases} \tag{4.4}$$

als Wert für $s(t)$ den von dem Punkt s^{i+1} aus zurückgelegten Weg erhält. Zur Bestimmung von v_{max} ist jedoch der absolute Weg nötig, also bestimme v_{max} als $v_{max}(s^{i+1} - s(t))$. Sonst funktioniert das Verfahren wie oben, man gibt sich Δt, ϵ und bricht ab, wenn $|\sigma(t)| < \epsilon$. Man erhält damit den Absprungort, die zugehörige Absprunggeschwindigkeit (entspricht der zulässigen Maximalgeschwindigkeit in diesem Punkt) sowie die zum Abbremsen benötigte Zeit.

Bemerkung 2 *Man erhält hier noch nicht den absoluten Zeitpunkt des Absprungs, dieser ergibt sich erst später.*

Der Fall, daß kein Absprungpunkt im Intervall $[s^i, s^{i+1}]$ existiert, obwohl die Geschwindigkeitsrestriktion im Punkt s^{i+1} nach unten springt, wird durch die anfangs angegebene Bedingung 1 an die Restriktionen ausgeschlossen.

4.3.1 Bestimmung des Absprungzeitpunktes

Wir haben in den letzten beiden Abschnitten Aufsprungort, Aufsprungzeitpunkt, Aufsprunggeschwindigkeit, Absprungort, Absprunggeschwindigkeit sowie Bremszeit bestimmt. Nun ist es also möglich, zu bestimmen, ob tatsächlich Auf- und Absprung vorliegen, oder ob nur ein Umschaltpunkt existiert. Falls der ermittelte Aufsprungort größer als der Absprungort ist, dann liegt ein Umschaltpunkt vor.
Vorerst sei jedoch der Aufsprungort kleiner als der Absprungort (und damit muß auch der zugehörige Aufsprungzeitpunkt kleiner sein als der Absprungzeitpunkt, da $s(t)$ eine stetige, monoton wachsende Funktion ist)
Was zur Fixierung des Absprungzeitpunktes noch fehlt, ist die Zeitdauer, mit der zwischen Auf- und Absprungort auf der Geschwindigkeitsrestriktion entlanggefahren wird. Auch diese Zeitdauer wird wieder als Nullstelle von folgender Funktion bestimmt:

$$\sigma(t) := s(t) - s_{ab} \tag{4.5}$$

Dabei ist s_{ab} der schon bestimmte Absprungort und $s(t)$ ist die Lösung des AWP

$$\begin{cases} \dot{s}(t) = v_{max}(s(t)) \\ s(t_{auf}) = s_{auf} \end{cases} \tag{4.6}$$

Dies entspricht der Fahrt auf der Geschwindigkeitsrestriktion $v_{max}(s(t))$, hierbei sind s_{auf} der Aufsprungort und t_{auf} die Aufsprungzeit.
Diese Nullstelle von $\sigma(t)$ sei t_{null}.
Es ergibt sich der Zeitpunkt des Absprungs zu $t_{ab} := t_{auf} + t_{null}$ und die Gesamtfahrzeit im Intervall ist also $t_{ges} := t_{ab} + t_{brems}$.
Damit sind alle Parameter der Schaltpunkte für den Fall des Auf- und Absprungs bestimmt.

4.4 Bestimmung des Umschaltpunktes

Wie schon im letzten Abschnitt dargestellt, gilt bei Vorliegen eines Umschaltpunktes, daß der ermittelte Absprungzeitpunkt vor dem Aufsprungzeitpunkt und analog der Absprungort vor dem Aufsprungort liegt.
Es existiert also ein Umschaltzeitpunkt t_{um} mit $t_{ab} \leq t_{um} \leq t_{auf}$ mit zugehörigem Ortspunkt s_{um}. Betrachtet man wieder die Geschwindigkeitstrajektorien der Vorwärtslösung des Anfangswertproblems (4.3) und die Lösung des Anfangswertproblems (4.4), dann ist deren Schnittpunkt gerade $v_1(t_{um}) = v_2(t_{um})$. Zusätzlich gilt in diesem Punkt, daß $s_1(t_{um}) = s_2(t_{um})$ gelten muß. (die Indizes 1 und 2 beziehen sich hierbei auf die entsprechenden Lösungen der beiden angegebenen AWP)
Numerisch wird folgendermaßen vorgegangen:

1. Gegeben seien die Intervallgrenzen s^i, s^{i+1}, die Anfangswerte t_i, v^i, die Restriktion im nächsten Intervallanfang $v^{i+1} = v_{max}(s^{i+1} + 0)$ sowie Genauigkeitsschranke ϵ und Startschrittweite Δt

2. Setze $t := \Delta t$

3. Löse das AWP (4.4) im Zeitpunkt t und bestimme den zugehörigen absoluten Ortspunkt s_2 und Geschwindigkeit v_2

4. Bestimme nun die Geschwindigkeit v_1, die die Lösung des AWP (4.3) in diesem Ortspunkt s_2 erreicht und die dafür benötigte Zeit t_1.

5. Berechne $\sigma := v_2 - v_1$

6. Wenn $|\sigma| < \epsilon$, dann setze :

   $$t_{um} := t_1$$
   $$t_{brems} := t$$
   $$s_{um} := s_1 (= s_2)$$
   $$v_{um} := v_1 (= v_2)$$

   und verlasse die Iteration

7. Wenn $\sigma < 0$, dann ist der Umschaltpunkt noch nicht erreicht, die Zeit t wird um Δt erhöht. (Bei Richtungswechsel Schrittweite Δt halbieren !)

8. Wenn $\sigma > 0$, dann ist der Umschaltpunkt überschritten, die Zeit t wird um Δt verringert. (Bei Richtungswechsel Schrittweite Δt halbieren !)

9. Gehe zu Punkt 3

Man erhält damit Zeit, Ort und Geschwindigkeit des Umschaltpunktes sowie die Bremszeit, die vom Umschaltpunkt aus bis zum Intervallende benötigt wird.
Die Gesamtfahrzeit beträgt also $t_{ges} = t_{um} + t_{brems}$.
In den Abbildungen 4.3, 4.4, 4.5 und 4.6 sind die s(t) und v(t) - Trajektorien für ein Beispiel mit Auf- und Absprungpunkt sowie ein Beispiel mit Umschaltpunkt dargestellt.

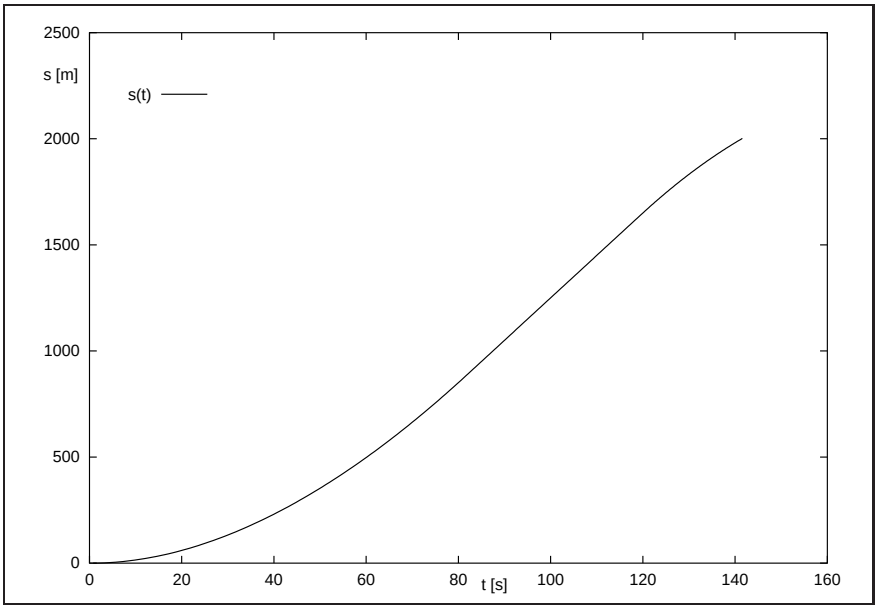

Abbildung 4.3: s-t Verlauf bei Auf- und Absprung

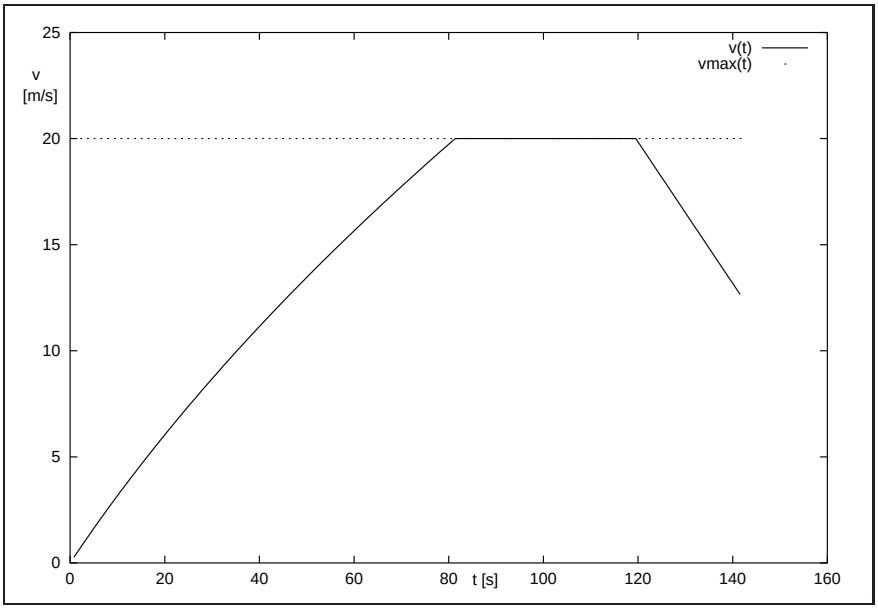

Abbildung 4.4: v-t Verlauf bei Auf- und Absprung

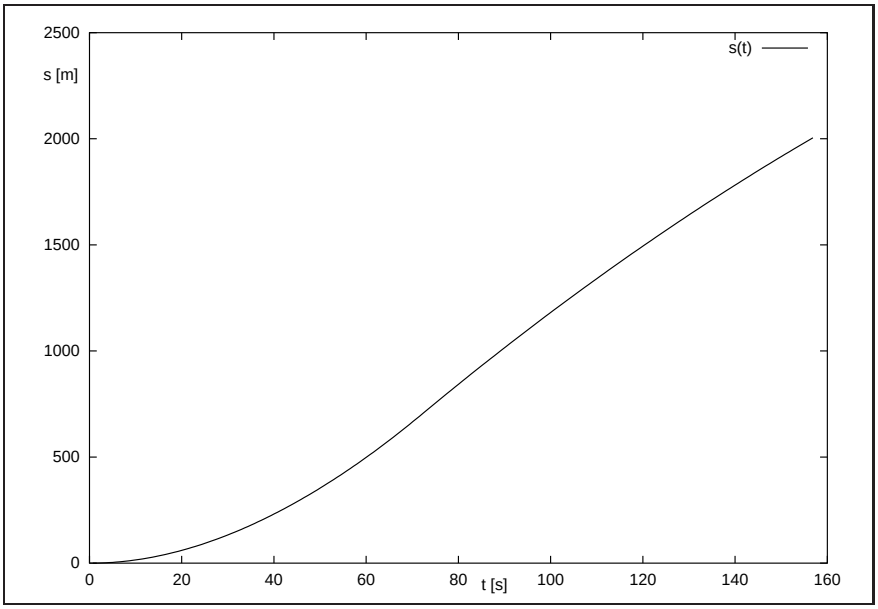

Abbildung 4.5: s-t Verlauf bei Umschaltpunkt

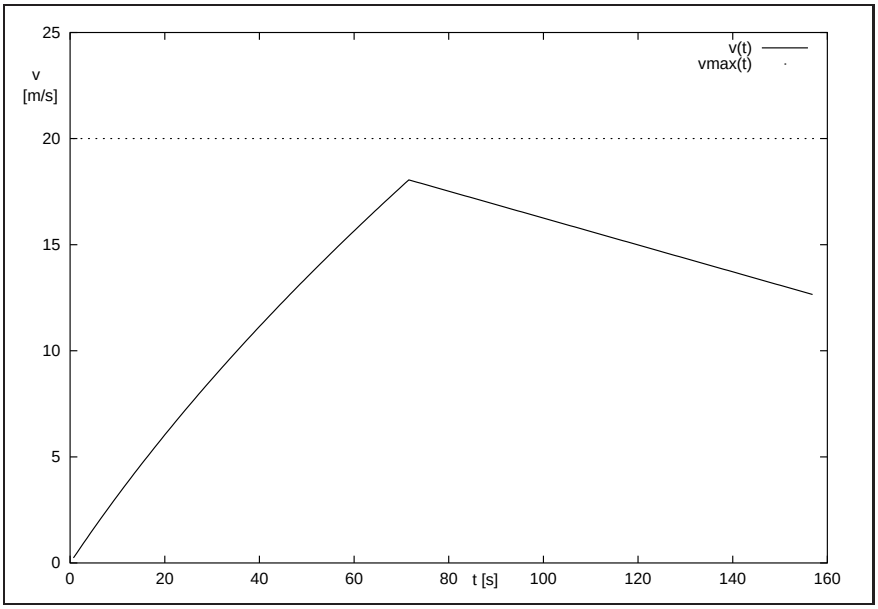

Abbildung 4.6: v-t Verlauf bei Umschaltpunkt

4.5 Variante zur Vermeidung der Bedingung 1

Da aber im praktischen Anwendungsfall bei der Deutschen Bahn die Streckenintervalle beliebig kurz sein können, kann die erste Bedingung nicht immer eingehalten werden. Deshalb ist es besser, folgende Strategie zur Bestimmung der Absprungpunkte zu verwenden:

1. Lege eine Liste mit allen rechtsseitigen Geschwindigkeitsbeschränkungen in den Punkten s^i an. Diese Werte seien in $\mathtt{v_{maxrechts}[i]}$ gespeichert. Ausserdem vermerke in der Liste, ob die Intervalle "normal "oder "zu kurz "sind.

2. Beginne im letzten Punkt der gesamten Strecke und versuche den Absprungpunkt s^n_{ab} in diesem letzten Intervall zu bestimmen. Falls er in diesem Intervall existiert, dann gehe weiter zum vorletzten Intervall und so weiter. Falls der Absprung in dem Intervall jedoch nicht zu schaffen ist, dann bestimme bei Erreichen der linken Intervallgrenze s^{i-1}, die in diesem Punkt gefahrene Geschwindigkeit und korrigiere die im Punkt 1 angelegte Liste mit den Maximalgeschwindigkeiten, indem $\mathtt{v_{maxrechts}[i-1]}$ auf diesen eben bestimmten Wert der Geschwindigkeit gesetzt wird.

   Für weitere Berechnungen wird nun dieses Intervall als „zu kurz" markiert. Dadurch weiß man bei der folgenden Bestimmung von Aufsprung- und Umschaltpunkten in derart markierten Intervallen, daß die Geschwindigkeitsrestriktion durch die Bremskurve dominiert wird.

Somit ergibt sich eine korrigierte zulässige Maximalgeschwindigkeit, die keine Abwärtssprünge mehr enthält, die sich auf weiter zurückliegende Intervalle auswirken können. Diese Idee und weitere praxisbedingte Besonderheiten sollen im folgenden bei der Entwicklung eines schnellen Verfahrens für die Anwendung zur Fahrzeitrechnung in den Betriebszentralen der Deutschen Bahn beachtet werden.

Kapitel 5

Das schnelle Verfahren für den praktisch relevanten Spezialfall

Um eine Berechnung der optimalen Steuerung und damit der Fahrzeiten des Zuges (daher der Name Fahrzeitrechnung) mit möglichst kurzer Laufzeit vorzunehmen, sind sämtliche Möglichkeiten der Vereinfachung durch spezielle Gestalten der Beschränkungen zu berücksichtigen.
Als Ergebnis ist das im folgenden beschriebene Verfahren zur Fahrzeitrechnung im Verlauf meines Praktikums bei der TLC-GmbH entstanden.

5.1 Ziel des Verfahrens

Ausgehend von den, durch das aufrufende Programm über einen template-Parameter `Iterator` übergebenen Daten der Streckenabschnitte ist für alle Abschnittsgrenzen die Fahrzeit sowie die Geschwindigkeit zu bestimmen. Dabei sollen folgende Bedingungen an die Eingangsdaten gelten:

- Innerhalb eines Abschnitts sind die zulässige Höchstgeschwindigkeit und die äußeren Kräfte sowie die Antriebskraft und maximale Bremsverzögerung bezüglich des Weges konstant.
- Die Antriebskraft und die äußeren Kräfte sind innerhalb eines Abschnitts stetig von der Geschwindigkeit abhängig.
- An den Abschnittsgrenzen sind Sprünge der Höchstgeschwindigkeit, der Bremsbeschleunigung und der äußeren Kräfte zulässig.

Das Verfahren durchläuft dafür die folgenden Schritte :

1. Anlegen einer Ereignisliste und Übernehmen der vorgegebenen Streckendaten
2. Anpassen der zulässigen Höchstgeschwindigkeit
3. Abfahren der Strecke
4. Postprozessing

die im weiteren einzeln beschrieben werden.

5.2 Anlegen der Ereignisliste

Zur Verwaltung der Ereignisse dient eine doppelt verkettete, lineare Liste, die Ereignispunkte entsprechend ihrer örtlichen Abfolge enthält. Ein Ereignispunkt umfaßt dabei die folgenden Informationen:

- `s` ... die örtliche Lage des Punktes.
- `fahrzeit` ... die (zu berechnende) Fahrzeit bis zum nächsten Ereignispunkt.
- `v` ... die (zu berechnende) Geschwindigkeit im Punkt.
- `vmax_links` ... die linksseitige zulässige Höchstgeschwindigkeit im Punkt.
- `vmax_rechts` ... analog die rechtsseitige zulässige Höchstgeschwindigkeit; falls das vorgegebene $\mathtt{v}_{\mathtt{max}}(\mathtt{s})$ stetig ist in `s`, dann ist `vmax_links` = `vmax_rechts`.
- `typ` ... die Art des Punktes, es wird zwischen `STRECKENPUNKT`, `AUFSPRUNGPUNKT` und `ABSPRUNGPUNKT` unterschieden. Streckenpunkte sind dabei die aus den übergebenen Streckenabschnittslängen ermittelten Anfangspunkte der Streckenintervalle. Aufsprung- und Absprungpunkte seien wie folgt definiert:

  Definition 6 (Aufsprung- und Absprungpunkte) *Sei $v_{max}(s)$ eine vorgegebene Geschwindigkeitsrestriktion und $v(s)$ eine Trajektorie einer Fahrt mit $v(s) \leq v_{max}(s) \quad \forall s$. Ein Punkt s ist ein Aufsprungpunkt, genau dann wenn $v(s-0) < v_{max}(s)$ und $v(s) = v_{max}(s)$ gilt.*

  Analog wird ein Punkt s mit $v(s+0) < v_{max}(s)$ und $v(s) = v_{max}(s)$ als Abbsprungpunkt bezeichnet.

- `streda` ... eine Kopie des vom rufenden Programm übergebenen Iterators um Zugriff auf die Strecken- und Fahrzeugdaten zu haben.
- `bremskurve` ... zur Kennzeichnung, ob im folgenden Intervall die zulässige Höchstgeschwindigkeit konstant (`bremskurve=0`) oder eine Bremskurve ist.

Zum Erstellen der Liste werden die Daten vom übergebenen Iterator übernommen und es ensteht eine Liste wie in Abbildung 5.1.

5.3 Anpassen der zulässigen Höchstgeschwindigkeit

5.3.1 Bestimmung von Absprungpunkten

An Stellen, in denen die zulässige Höchstgeschwindigkeit nach unten springt, müssen Absprungpunkte bestimmt werden, da sonst mit einer endlichen maximalen Bremsverzögerung die zulässige Höchstgeschwindigkeit im Sprungpunkt nicht eingehalten werden kann. Dazu wird am Ende der Ereignisliste begonnen und falls in einem Punkt die Sprungbedingung

$$\mathtt{vmax\_links} > \mathtt{vmax\_rechts} \tag{5.1}$$

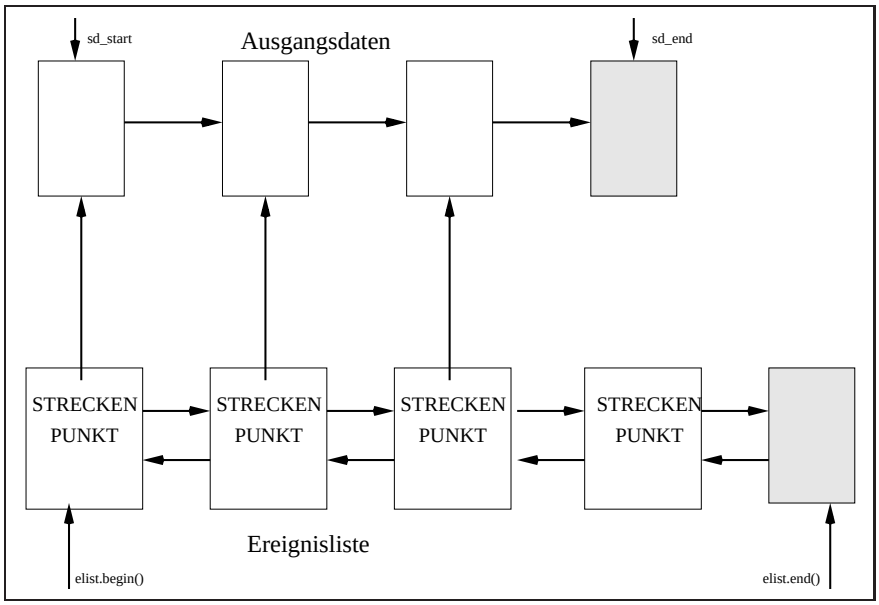

Abbildung 5.1: Ereignisliste nach Übernahme der Streckendaten

gilt, dann muß ein Absprungpunkt bestimmt werden oder die bisher konstante Höchstgeschwindigkeit in dem Intervall vor dem Sprungpunkt durch eine Bremskurve ersetzt werden. Sei also wie in Abbildung 5.2 $\tilde{s}$ der Punkt in dem $v_{max}(s)$ von $\hat{v}$ auf $\tilde{v}$ nach unten springt. Außerdem sei $\hat{s}$ der Anfang des Intervalls und s^* der zu bestimmende Absprungpunkt.
Die Bremsverzögerung a ist als konstant vorausgesetzt, also gilt bei einem Start in s^* zur Zeit t_0=0 mit Geschwindigkeit $\hat{v}$:

$$v(t) = at + \hat{v} \tag{5.2}$$

$$s(t) = \frac{a}{2}t^2 + \hat{v}t + s^* \tag{5.3}$$

Das Ziel ist, auf die Geschwindigkeit $v(t) = \tilde{v}$ abzubremsen, also ergibt sich aus der ersten Gleichung die dafür benötigte Zeit zu

$$t^* = \frac{\tilde{v} - \hat{v}}{a} \tag{5.4}$$

und Umstellen der zweiten Gleichung ergibt den gesuchten Absprungpunkt

$$s^* = -\frac{a}{2}(t^*)^2 - \hat{v}t^* + \tilde{s}. \tag{5.5}$$

Wenn nun gilt, daß $s^* > \hat{s}$, dann haben wir einen Absprungpunkt gefunden. Es wird ein neuer Ereignispunkt vom Typ ABSPRUNGPUNKT erzeugt, s^* als Ort des Punktes angenommen, die Eigenschaft Bremskurve auf 1 gesetzt und die Streckendaten durch Kopieren des Iterators vom vorherigen Punkt (am Ort $\hat{s}$) übernommen. Zum Schluß wird der neue Punkt in die Ereignisliste eingeordnet.

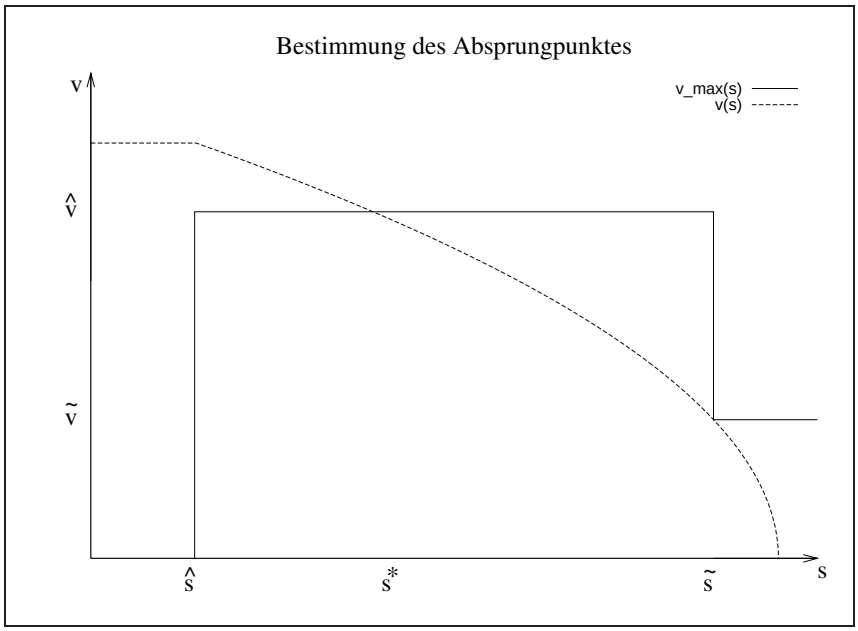

Abbildung 5.2: Bestimmen des Absprungpunktes

5.3.2 Korrektur von $v_{max}(s)$ in „zu kurzen“ Intervallen

Es kann aber auch passieren, daß das Intervall zu kurz ist, d.h. $s^* \leq \hat{s}$. In diesem Fall wird am Anfang des Intervalls `vmax_rechts` auf den entsprechenden Wert v^* nach unten korrigiert. Gesucht ist also eine passende Startgeschwindigkeit v^* im Punkt $\hat{s}$ bei der sich mit konstanter Bremsbeschleunigung a im Punkt $\tilde{s}$ die Geschwindigkeit $\tilde{v}$ ergibt (siehe auch Abbildung 5.3) .
Wir haben wieder die beiden Bewegungsgleichungen

$$v(t) = at + v^* \tag{5.6}$$
$$s(t) = \frac{a}{2}t^2 + v^*t + \hat{s}. \tag{5.7}$$

Aus der ersten Gleichung erhält man v^* in Abhängigkeit von der (unbekannten) Fahrzeit

$$v^* = \tilde{v} - at. \tag{5.8}$$

Eingesetzt in die zweite Gleichung ergibt sich über

$$\tilde{s} = \frac{a}{2}t^2 + (\tilde{v} - at)t + \hat{s} \tag{5.9}$$

eine quadratische Gleichung zur Bestimmung der Bremszeit:

$$0 = t^2 - \frac{2\tilde{v}}{a}t - \frac{2(\hat{s} - \tilde{s})}{a} \tag{5.10}$$

Die beiden Lösungen für t sind

$$t_{1/2} = \frac{\tilde{v}}{a} \pm \sqrt{\frac{\tilde{v}^2}{a^2} + \frac{2(\hat{s} - \tilde{s})}{a}}. \tag{5.11}$$

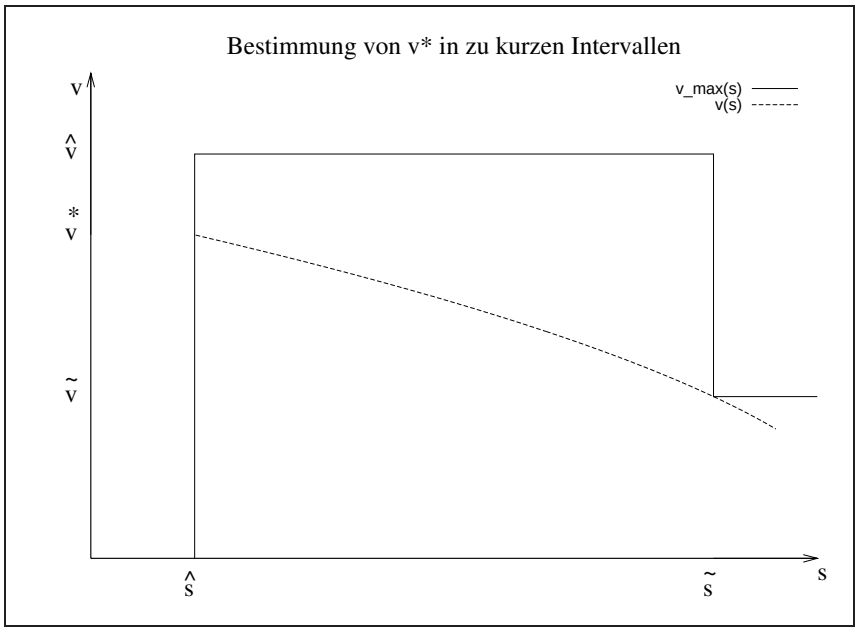

Abbildung 5.3: Korrektur von v_{max} in zu kurzen Intervallen

Dabei ergibt sich mit $\frac{\tilde{v}}{a} + \sqrt{\ldots}$ das richtige (positive) t nach folgender Abschätzung:

Haben $a < 0$ und $\tilde{s} > \hat{s}$ also gilt,

$$\frac{2(\hat{s} - \tilde{s})}{a} > 0 \tag{5.12}$$

und es ist erst einmal gesichert, daß die Wurzel reell ist. Weiterhin folgt aus dieser Abschätzung

$$\sqrt{\frac{\tilde{v}^2}{a^2} + \frac{2(\hat{s} - \tilde{s})}{a}} > \sqrt{\frac{\tilde{v}^2}{a^2}} = \frac{\tilde{v}}{a} \tag{5.13}$$

woraus wiederum folgt, daß

$$\frac{\tilde{v}}{a} - \sqrt{\frac{\tilde{v}^2}{a^2} + \frac{2(\hat{s} - \tilde{s})}{a}} < 0 \tag{5.14}$$

und es ergibt sich als das gesuchte t

$$t = \frac{\tilde{v}}{a} + \sqrt{\frac{\tilde{v}^2}{a^2} + \frac{2(\hat{s} - \tilde{s})}{a}}. \tag{5.15}$$

Nachdem man so die Bremszeit bestimmt hat, ergibt sich aus Gleichung (5.8) die gesuchte Startgeschwindigkeit v^*, die auf `vmax_rechts` im Startpunkt des Intervalls kopiert wird. Außerdem wird in diesem Intervall `bremskurve` auf 1 gesetzt, da die zulässige Höchstgeschwindigkeit nicht mehr konstant ist.

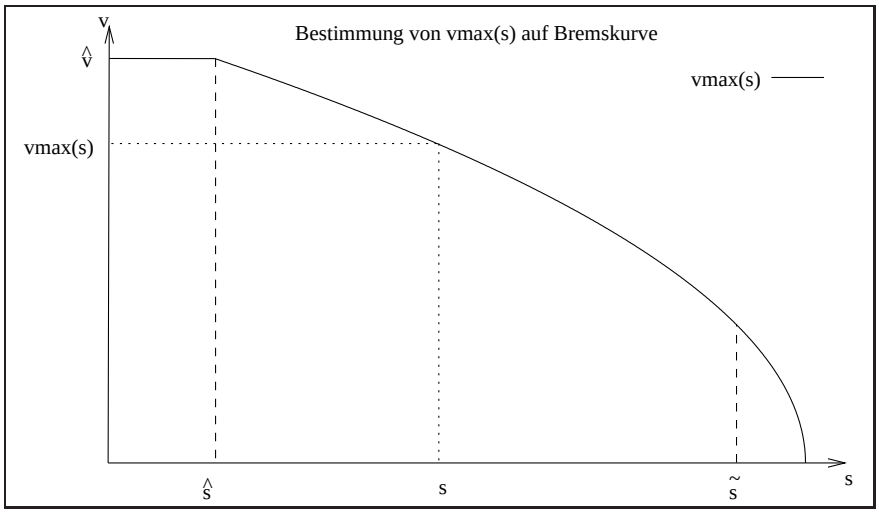

Abbildung 5.4: $v_{max}(s)$ als Bremskurve

5.3.3 Bestimmung von $v_{max}(s)$ in Intervallen mit Bremskurven

In Intervallen mit Bremskurven ist $v_{max}(s)$ ein Parabelbogen wie in Abbildung 5.4. Bekannt sind im Intervall der Startpunkt $\hat{s}$ mit der zugehörigen Höchstgeschwindigkeit $\hat{v}$ = `vmax_rechts` sowie die Bremsverzögerung a. Man leitet sich also wieder wie im letzen Abschnitt aus den beiden Bewegungsgleichungen

$$v(t) = at + \hat{v} \tag{5.16}$$
$$s(t) = \frac{a}{2}t^2 + \hat{v}t + \hat{s} \tag{5.17}$$

die Beziehung für die Fahrzeit $t(s)$ her als

$$t(s) = -\frac{\hat{v}}{a} + \sqrt{\frac{\hat{v}^2}{a^2} - \frac{2(\hat{s}-s)}{a}} \tag{5.18}$$

und setzt dies in $v(t) = at + \hat{v}$ ein, um die $v(s)$ - Beziehung

$$v(s) = a\left[-\frac{\hat{v}}{a} + \sqrt{\frac{\hat{v}^2}{a^2} - \frac{2(\hat{s}-s)}{a}}\right] + \hat{v} \tag{5.19}$$
$$= -\hat{v} + a\sqrt{\frac{\hat{v}^2}{a^2} - \frac{2(\hat{s}-s)}{a}} + \hat{v} \tag{5.20}$$
$$= a\sqrt{\frac{\hat{v}^2}{a^2} - \frac{2(\hat{s}-s)}{a}} \tag{5.21}$$
$$= \sqrt{\hat{v}^2 - 2a(\hat{s}-s)} \tag{5.22}$$

zu erhalten.
Zu beachten ist, daß bei zu großem s die Wurzel nicht mehr reell ist, das kann vor allem auf Bremskurven, die bis zu $v_{max}(s) = 0$ gehen, durch Rundungsfehler eintreten.

Im numerischen Verfahren wird dies überwacht, und im Falle eines negativen Radikanten als $v_{max}(s)$ der Wert Null zurückgegeben.
In den beiden Abbildungen 5.5 und 5.6 sind die Unterschiede im Verlauf von $v_{max}(s)$ vor und nach der Korrektur zu sehen.
Im weiteren sei mit $v_{max}(s)$ immer die korrigierte Maximalgeschwindigkeit bezeichnet.

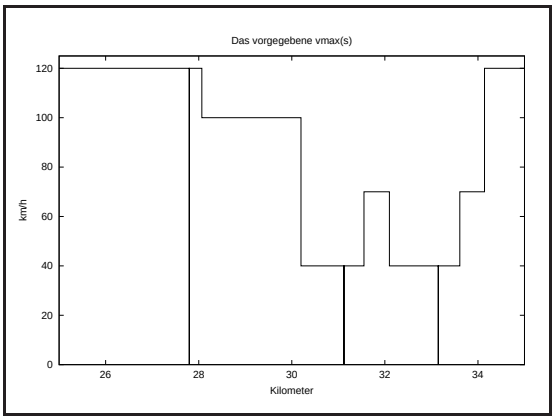

Abbildung 5.5: $v_{max}(s)$ vor der Korrektur

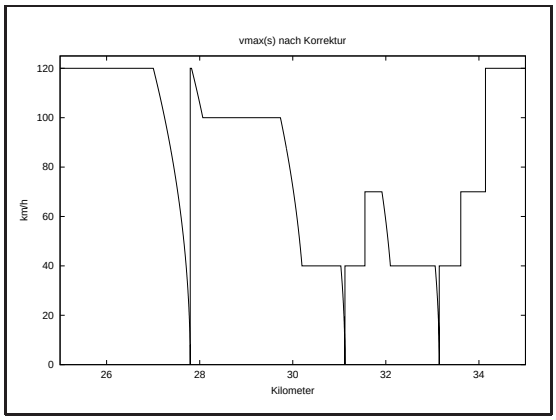

Abbildung 5.6: $v_{max}(s)$ nach der Korrektur

5.4 Abfahren der Strecke

Die Strecke wird intervallweise von vorn nach hinten abgefahren, Startwerte im ersten Intervall sind Startzeit $t_0 = 0$ und die vom aufrufenden Programm übergebene Startgeschwindigkeit v_0.
In jedem Intervallanfang wird zwischen einem freien Start und einem Start auf der Beschränkung unterschieden.

5.4.1 Start auf Beschränkung

Ein Start auf der Geschwindigkeitsbeschränkung liegt vor, wenn im Intervallanfang s die Startgeschwindigkeit v gleich der Maximalgeschwindigkeit $v_{max}(s)$ ist und außerdem die Antriebskraft $u_{max}(v)$ stark genug ist, die äußeren Kräfte $u_f(v)$ zu kompensieren, d.h. $u_{max}(v) \geq -u_f(v)$.
In diesem Fall wird das Intervall mit der zulässigen Maximalgeschwindigkeit durchfahren, da die äußeren Kräfte $u_f(v)$ innerhalb des Intervalls bezüglich s konstant sind und $v_{max}(s)$ entweder konstant ist oder fällt.
Im Falle $v_{max}(s) = const$ ergibt sich die Fahrzeit für das Intervall zu $t = \frac{\texttt{intervallaenge}}{v_{max}}$.
Falls $v_{max}(s)$ nicht konstant ist, so ergibt sich aus der Beziehung für Bewegung mit konstanter Beschleunigung $v_{neu} = at + v_{alt}$ die Fahrzeit zu $t = \frac{v_{neu} - v_{alt}}{a}$. Dabei sei $v_{alt} = v_{max}(s)$ und $v_{neu} = v_{max}(s + \texttt{intervallaenge})$.
Die Startgeschwindigkeit im nächsten Intervall ist in beiden Fällen gleich der linksseitigen Maximalgeschwindigkeit `vmax_links` im nächsten Punkt.

5.4.2 Freier Start

Im Falle des freien Starts gilt am Intervallanfang $v \in [0, v_{max})$. Es ist also das Anfangswertproblem mit der Bewegungsdifferentialgleichung

$$\begin{aligned} \dot{s} &= v && (5.23)\\ \dot{v} &= \frac{1}{m}(u_{max}(v) + u_f(v)) && (5.24)\\ s(0) &= s_0 && (5.25)\\ v(0) &= v_0 && (5.26) \end{aligned}$$

solange zu lösen, bis $s(t)$ größer als die rechtsseitige Intervallgrenze oder $v(s(t))$ größer als die zulässige Höchstgeschwindigkeit $v_{max}(s(t))$ wird.
Als Differentialgleichungslöser sind drei verschiedene Verfahren implementiert, nämlich Euler-Vorwärts mit fester Schrittweite, Euler-Vorwärts mit Schrittweitensteuerung sowie Runge-Kutta 4. Ordnung mit Schrittweitensteuerung.
Einzelheiten zu diesen Verfahren sind z. B. in [10] zu finden.
Zur Schrittweitensteuerung wird in jedem Schritt die Differentialgleichung einmal mit ganzer Schrittweite und zweimal mit halber Schrittweite gelöst, die Schrittweite wird akzeptiert, wenn die beiden so erhaltenen Lösungen für $v(t)$ entsprechend nah beieinander liegen. Als Steuerparameter dient dabei der Parameter `eps_v`.
Wird eine der oben angegebenen Abbruchbedingungen erreicht, dann werden die Werte für s bzw. v wie folgt interpoliert. (Dabei seien t_{alt}, s_{alt}, v_{alt} die entspechenden Werte des vorletzten Lösungsschrittes und t_{neu}, s_{neu}, v_{neu} die Werte des letzen Schrittes sowie s_{rechts} die rechtsseitige Intervallgrenze.)

Lineare Interpolation

Fall 1: $v_{neu} > v_{max}(s_{neu})$
Die zulässige Höchstgeschwindigkeit wird überschritten, man erhält also einen Aufsprungpunkt.
Der Ort s^*dieses Punktes wird durch den Schnittpunkt zwischen den Geraden der Linearisierungen von der Bewegung $v(s)$ und der Restriktion $v_{max}(s)$ bestimmt.
Wir haben also für die linearisierte Bewegungsgleichung

$$v(s) = v_{alt} + \frac{v_{neu} - v_{alt}}{s_{neu} - s_{alt}}(s - s_{alt}) \tag{5.27}$$

und für die Restriktion

$$v_{max}(s) = v_{max}(s_{alt}) + \frac{v_{max}(s_{neu}) - v_{max}(s_{alt})}{s_{neu} - s_{alt}}(s - s_{alt}). \tag{5.28}$$

Gesucht ist der Punkt s, an dem gilt $v_{max}(s) = v(s)$, mit den beiden Hilfsgrößen

$$\begin{aligned} \alpha &:= \frac{v_{neu} - v_{alt}}{s_{neu} - s_{alt}} && (5.29)\\ \beta &:= \frac{v_{max}(s_{neu}) - v_{max}(s_{alt})}{s_{neu} - s_{alt}} && (5.30) \end{aligned}$$

ergibt sich

$$v(s) = v_{alt} + \alpha(s - s_{alt}) \tag{5.31}$$
$$v_{max}(s) = v_{max}(s_{alt}) + \beta(s - s_{alt}) \tag{5.32}$$

und man erhält durch Gleichsetzen von (5.31) mit (5.32) und Umstellen das gesuchte s^* zu

$$s^* = \frac{v_{alt} - v_{max}(s_{alt})}{\beta - \alpha} + s_{alt}. \tag{5.33}$$

Die zugehörige Zeit wird auch linear interpoliert zu

$$t^* = t_{alt} + \frac{t_{neu} - t_{alt}}{s_{neu} - s_{alt}}(s^* - s_{alt}). \tag{5.34}$$

Fall 2: $s_{neu} > s_{rechts}$
In diesem Fall wird das Intervall frei durchfahren, ohne einen Aufsprungpunkt zu erhalten. Es ist also die Austrittsgeschwindigkeit v^* aus dem Intervall und die zugehörige Zeit t^* zu bestimmen.
Man erhält diese leicht durch Einsetzen von $s = s_{rechts}$ in (5.27) und (5.34) als

$$v^* = v_{alt} + \frac{v_{neu} - v_{alt}}{s_{neu} - s_{alt}}(s_{rechts} - s_{alt}) \tag{5.35}$$
$$t^* = t_{alt} + \frac{t_{neu} - t_{alt}}{s_{neu} - s_{alt}}(s^* - s_{alt}) \tag{5.36}$$

Als Sonderfall ist dabei noch zu beachten, daß auch beide Abbruchbedingungen erfüllt sein können. Dann kann die Behandlung des Falls 2 zu einem v^* führen, welches größer als $v_{max}(s_{rechts})$ ist. Falls das eintritt, wird im Verfahren diese Lösung verworfen und der Fall wie der Fall 1 behandelt.

Polynomiale Interpolation

Testrechnungen zeigen, daß die Genauigkeit bei linearer Interpolation nicht sehr gut ist. Günstiger ist, eine polynomiale Interpolation vorzunehmen.
Gut geeignet ist dafür die Newton-Interpolation über dividierte Differenzen [5],[6].
Qualitativ unterscheiden sich die beiden obigen Fälle dadurch, daß im zweiten Fall nur die Interpolationspolynome für $v(s)$ und $t(s)$ aufgebaut und an der rechten Intervallgrenze ausgewertet werden müssen, während beim ersten Fall eine Nullstellenbestimmung nötig ist. Dafür geeignet ist das Newtonverfahren.[6]
Zum Aufbau der Polynome braucht man neben den Werten des letzten und vorletzten Lösungsschrittes der DGL je nach gewählten Polynomgrad weitere Stützstellen. Diese werden durch zusätzliche Auswertungen der DGL an Zwischenpunkten gewonnen. Hat

man diese Werte bestimmt, so lassen sich die Koeffizienten a_k des Interpolationspolynoms

$$\begin{aligned} P(x) &= a_0 \\ &\quad +a_1(x-x_0) \\ &\quad +a2(x-x_0)(x-x_1)+ \\ &\quad +\dots \\ &\quad +a_n(x-x_0)(x-x_1)\dots(x-x_{n-1}) \qquad (5.37) \\ &= (\dots(a_n(x-x_{n-1})+a_{n-1})(x-x_{n-2})+\dots+a_1)(x-x_0)+a_0 \qquad (5.38) \end{aligned}$$

folgendermaßen schnell ausrechnen

```
for (i=0;i<=n;i++)
   {
   a[i]=f[i];
   for (k=0;k<=i-1;k++)
      {
      a[i]=(a[i]-a[k]) / (x[i]-x[k]);
      }
   }
```

Dabei sind `a[i]` die gesuchten Koeffizienten, `x[i]` die Stützstellen, `f[i]` die Funktionswerte in den Stützstellen und `n` der Polynomgrad.
Zur Auswertung des Interpolationspolynoms an einer beliebigen Stelle `phi`, die aber innerhalb des Intervalls der Stützstellen liegen sollte ist der folgende, dem Horner-Schema [6],[5] abgeschaute Algorithmus effektiv.

```
wert=a[n];
for (i=n-1;i>=0;i--)
   wert=wert*(phi-x[i]) + a[i];
```

Benötigt man, wie im Falle des Newton-Verfahrens, außer dem Wert des Polynoms auch seine Ableitung, so ist es besser, den folgenden Algorithmus zu verwenden, da er beide Werte in einem Durchlauf liefert

```
wert=a[n];
ablt=a[n];
for (i=n-1;i>0;i--)
   {
   wert=wert*(phi-x[i])+a[i];
   ablt=ablt*(phi-x[i-1])+wert;
   }
wert=wert*(phi-x[0])+a[0];
```

Damit lassen sich die Werte für $v($`srechts`$)$ und $t($`srechts`$)$ durch Auswertung der entsprechend aufgebauten Polynome an der Stelle `srechts` gewinnen. Für die Bestimmung des Aufsprungpunktes auf die Restriktion $v_{max}(s)$ ist der Schnittpunkt von $v_{max}(s)$ mit dem Interpolationspolynom $v_p(s)$ von $v(s)$ zu berechnen.

Dafür wird noch die Ableitung von $v_{max}(s)$ gebraucht. Wir hatten in (5.22) $v_{max}(s)$ in Intervallen mit Bremskurven als

$$v_{max}(s) = \sqrt{\hat{v}^2 - 2a(\hat{s} - s)} \tag{5.39}$$

hergeleitet, sonst ist $v_{max}(s)$ konstant. Somit ergibt sich als Ableitung von $v_{max}(s)$

$$v'_{max}(s) = \begin{cases} 0 & \text{falls } \texttt{bremskurve = 0} \\ \frac{a}{\sqrt{\hat{v}^2 - 2a(\hat{s}-s)}} & \text{sonst} \end{cases} \tag{5.40}$$

Es kann nun der Schnittpunkt von $v_{max}(s)$ und $v_p(s)$ als Nullstelle von

$$\sigma(s) := v_{max}(s) - v_p(s) \tag{5.41}$$

bestimmmt werden.
Im Verfahren wird dies mit dem Newtonverfahren durchgeführt. Als Startwert wird der Wert, der sich durch lineare Interpolation ergibt, gewählt und anschließend wird dieser Wert s^* durch

$$s^* := s^* - \frac{\sigma(s)}{\sigma'(s)} \tag{5.42}$$

iterativ verbessert, bis die Abbruchschranke $|\sigma(s^*)| < \texttt{eps\_v}$ oder die maximal zulässige Anzahl von Iterationen erreicht ist.
Die zugehörige Aufsprungzeit kann nun durch Auswertung des entsprechenden Interpolationspolynoms für $t(s)$ an der eben errechneten Stelle s^* bestimmt werden.

5.4.3 Sonderfälle

Es können noch folgende Sonderfälle auftreten:

Halteintervalle

In Halteintervallen gilt, daß die Länge des Intervalls gleich Null ist. Die vorgegebene Haltezeit wird als Fahrzeit angenommen und im nächsten Intervall mit $v = v_{max}(s) = 0$ gestartet.

Intervalle mit Länge Null

Um Ausgaben an Meßpunkten zu ermöglichen, kann es entstehen, daß Intervalle die Länge Null haben. Dies würde in der Numerik zu Abstürzen wegen Division durch Null führen. Solche Intervalle werden vorher abgefangen, ihre Fahrzeit wird auf Null gesetzt und die Startgeschwindigkeit dieses Intervalls wird als Startgeschwindigkeit des nächsten Intervalls angenommen. Um zu entscheiden, ob ein Intervall ein Null-Intervall ist, wird die Länge des Intervalls mit dem Steuerparameter `eps_s` verglichen und wenn gilt, daß `intervallaenge` $<$ `eps_s`, als Null angenommen.

Intervalle mit Aufsprungpunkt als Startpunkt

Aufsprungpunkte werden erst während der Fahrt eingefügt und das Reststück des Intervalls bereits beim Finden des Aufsprungpunktes in die Fahrzeit einbezogen. Deshalb werden solche Punkte nicht als Startpunkte beachtet und mit dem nächsten Punkt weitergearbeitet.

Zu große Steigungen

Falls die Steigung der Strecke zu groß wird, kann es dazu kommen, daß eine Weiterfahrt des Zuges mangels Antriebskraft nicht möglich ist. Dieser Fall wird während der schrittweisen Lösung der Bewegungsdifferentialgleichung durch Überprüfung der errechneten Geschwindigkeit überwacht. Wenn die eben bestimmte Geschwindigkeit kleiner als Null ist, dann wird das Verfahren mit einer entsprechenden Exception abgebrochen.

5.5 Postprozessing

Nachdem die Strecke abgefahren wurde und somit in allen Punkten der Ereignisliste Fahrzeit und Geschwindigkeit bekannt sind, wird die Fahrzeit in den Streckenpunkten mit und ohne Regelzuschlägen aufsummiert und an das aufrufende Programm zurückgegeben.

5.6 Numerische Steuerparameter

Zu Testzwecken können an das Verfahren verschiedene Steuerparameter übergeben werden und es ist möglich verschiedene Varianten mit und ohne Ausgaben durch Definition bestimmter Makros zu erzeugen.
Wird das Makro `FZR_TRACE_LOGIK` definiert, so werden im Verzeichnis ./aus Gnuplot-Datenfiles zum Plotten von v-s, t-s, vmax-s - Diagrammen, Ausgaben der Ereignisliste und anderes mehr erzeugt. Darstellbar sind die Plots durch Starten von Gnuplot im Verzeichnis ./aus und Aufruf von load “plotall“ .
Definition des Makros `ZEITAUSGABE` bewirkt eine Laufzeitmessung in den einzelnen Teilen des Verfahrens und Ausgabe am Schluß. Zeitmessung ist nur effektiv, wenn keine Ausgaben erzeugt werden, da sonst zum größten Teil die Zeit für Dateioperationen verbraucht wird.
Weiterhin können über eine Steuerdatei der folgenden Form

```
#
# Numerik-Parameter fuer pk_rechne
#
# verfahren     ... Art des DGL-Solver
#         0     ... Euler mit fester SW (=delta_t_start)
#         1     ... Euler mit SW-Steuerung
#         2     ... Runge-Kutta mit SW-Steuerung
#         3     ...   "     "   mit besserer SW-Steuerung
#         4     ... Euler       mit  "    "    "    "
#
```

```
# eps_v         ... Genauigkeitsschranke fuer SW-Steuerung
#
# eps_s         ... Schranke, ab der Intervalle als
#                   Null-Laengen-Intervalle angenommen werden
#
# delta_t_start ... Startschrittweite fuer DGL -Loeser
#
# eps_srechts   ... Maximal gestattetes Ueberschreiten der
#                   rechten Intervallgrenze
#
# delta_t_max   ... Maximale Schrittweite
#                   ( z.Zt. nur bei Verfahren 3 und 4 )
#
#
# polynomgrad   ... Polynomgrad zur Interpolation <= 9 !!!!!
#
verfahren: 3
eps_v:  1.0e-3
eps_s: 0.001
delta_t_start: 0.01
eps_srechts: 1.0
delta_t_max: 2000.0
polynomgrad: 2
```

Steuerparameter an das Verfahren übergeben werden.

5.7 Handhabung des Programms

5.7.1 Erstellen und Verwendung verschiedener Versionen

Die Handhabung des Programms erfolgt analog zur schon vorhandenen Testumgebung. Darüber hinaus lassen sich durch Verwendung verschiedener Makros beim Compilieren verschiedene Versionen erstellen.
Prinzipiell sind die Makros

`FZR_TRACE_LOGIK`, `ZEITAUSGABE` und `TEST_UMAX_HYPERBEL`

vorgesehen.
Ist `FZR_TRACE_LOGIK` definiert, dann werden im Verzeichnis `./aus` Plot-Dateien erzeugt, die eine Darstellung der Ergebnisse mittels Gnuplot ermöglichen.
Ist `ZEITAUSGABE` definiert, dann erfolgt in den einzelnen 4 Abschnitten des Programms eine Laufzeitmesseung und Ausgabe der benötigten Zeiten am Schluss. Zeitmessungen sind nur sinnvoll, wenn keine Ausgaben erfolgen. Außerdem sollte das Programm dann mit der Optimierungsoption `-O2` erstellt werden.

Das Makro `TEST_UMAX_HYPERBEL` erstellt eine spezielle Version des Programms für einen Vergleich mit einer analytischen Lösung. Dabei werden die Zugkraftkurve und die äußeren Kräfte wie in den Testrechnungen angegeben geändert.
Um die Erstellung dieser Versionen zu erleichtern, existieren die Shellscripts

`makeall-roman.sh` und `makeana-roman.sh`

die folgende Versionen erstellen:

`praktikum_testumgebung_optimiert2` ... optimierte Version mit `ZEITAUSGABE`

`praktikum_testumgebung_ausgabe` ... Version mit `FZR_TRACE_LOGIK`

`praktikum_testumgebung_ana` ... Version mit `TEST_UMAX_HYPERBEL`

Zum Aufruf eines Testfalls existieren die Scripte

`do-test-roman.pl` und `do-test-roman-schnell.pl`

die die entsprechenden Versionen des Programms verwenden und sonst analog zu den Originalscripten der Testumgebung gehandhabt werden.

5.7.2 Rechnen von Testreihen

Zur Durchführung und Dokumentation von Testreihen mit verschiedenen Parametern existieren die Verzeichnisse

`./uro/analytisch1` und `./uro/realtest` .

In diesen lässt sich jeweils das Script `maketest.sh` ausführen, welches die Testreihen rechnet und die Ausgaben in entsprechenden Unterverzeichnissen bereitstellt.
Der Spielraum der Parameter ist dabei im File `gen_para.c` anzugeben.
Anschließend können die Tabellen im LaTeX - Stil mit `makeausgabe.sh` erzeugt werden, sie entstehen im Unterverzeichnis `latex`.

5.8 Testrechnungen

Im weiteren folgen einige Testbeispiele.

5.8.1 Ein analytisch berechenbarer Testfall

In diesem Testfall wird als maximale Zugkraft eine Hyperbel angenommen, um so eine geschlossene Lösung zur Bestimmung eines Aufsprungpunktes und der Fahrzeit zu erhalten.
Betrachten wir wieder die Bewegungs-Differentialgleichung

$$\dot{s} = v \tag{5.43}$$

$$\dot{v} = \frac{1}{m} u(v) \tag{5.44}$$

wobei in $u(v)$ die äußeren Kräfte mit Null und die Antriebskraft als Hyperbel angenommen wird. Wir verwenden im weiteren als $u(v)$

$$u(v) = \frac{250000}{v} \tag{5.45}$$

und es ergibt sich für $\dot{v}$

$$\begin{aligned} \dot{v} &= \alpha\frac{1}{v} \qquad \text{mit} \quad \alpha := \frac{250000}{m} && (5.46)\\ v\dot{v} &= \alpha \qquad v\dot{v} = \frac{1}{2}(\dot{v^2}) \quad w := v^2 && (5.47)\\ \dot{w} &= 2\alpha && (5.48)\\ w &= 2\alpha t + C_1 && (5.49) \end{aligned}$$

und damit für $v(t)$

$$v(t) = \sqrt{2\alpha t + C_1} \tag{5.50}$$

wodurch sich aus

$$\dot{s} = v = \sqrt{2\alpha t + C_1} \tag{5.51}$$

$s(t)$ ergibt zu

$$s(t) = \frac{2}{3}\frac{(2\alpha t + C_1)^{\frac{3}{2}}}{2\alpha} + C_2. \tag{5.52}$$

Wir wählen als Anfangswerte $s(0) = 0$ und $v(0) = 1$ wodurch sich für die Konstanten $C_1 = 1$ und $C_2 = -\frac{2}{6\alpha}$ ergibt.
Außerdem sei die Zugmasse m gleich 500000 kg, daraus folgt $\alpha = \frac{1}{2}$ und wir erhalten

$$\begin{aligned} s(t) &= \frac{2}{3}(t+1)^{\frac{3}{2}} - \frac{2}{3} && (5.53)\\ v(t) &= \sqrt{t+1} \quad . && (5.54) \end{aligned}$$

Startet man nun in $t = 0$ mit $s(0) = 0$ und $v(0) = 1$, so wird eine vorgegebene Maximalgeschwindigkeit von 30 m/s nach 899 Sekunden erreicht, dabei werden 17999.33 Meter zurückgelegt.

$$\begin{aligned} v(899) &= \sqrt{899+1} && (5.55)\\ &= 30 && (5.56)\\ s(899) &= \frac{2}{3}(899+1)^{\frac{3}{2}} - \frac{2}{3} && (5.57)\\ &= 17999.33 \quad . && (5.58) \end{aligned}$$

Es ergeben sich also folgende Parameter, die der Numerik übergeben werden

m=500000 kg

$v(0)$=1 m/s

$vmax$=30 m/s

und es müßte im Idealfall ein Aufsprungpunkt zur Zeit 899 Sekunden am Ort 17999.33 Meter gefunden werden.
Analog dazu konstruiert man ein Beispiel, in dem ein Intervall der Länge 5332.67 frei durchfahren und im Idealfall zur Zeit $t = 399$ Sekunden mit Geschwindigkeit $v = 20.0$ m/s verlassen werden sollte, indem man sich die Zeit mit $t = 399$ Sekunden vorgibt, und aus

$$\begin{aligned} v(399) &= \sqrt{399+1} && (5.59)\\ &= 20 && (5.60)\\ s(399) &= \frac{2}{3}(399+1)^{\frac{3}{2}} - \frac{2}{3} && (5.61)\\ &= 5332.67 && (5.62) \end{aligned}$$

die beiden Werte bestimmt.
Im Anhang folgen auf den Seiten 111 ff. einige Tabellen zum Aufwand und Fehler in Abhängigkeit vom Polynomgrad der Interpolation bei Testrechnungen zu diesem Beispiel.

5.8.2 Testfall 22127-1013

Das folgende Beispiel wurde auf einem Intel Pentium 2, 450 MHz, 384 MB RAM gerechnet. Es sind 80 Kilometer Strecke mit 1275 Intervallen abzufahren.
Die Nummer 22127-1013 bezieht sich dabei auf die Bezeichnungen der Datenkonzeption in den Betriebszentralen der Bahn, in diesem Falle die BZ Niederlassung Mitte.
Dabei ergeben sich folgende Rechenzeiten in den einzelnen Teilen

Liste anlegen	Absprungpunkte bestimmen	Strecke abfahren	Postprozessing
0.04 s	0.01 s	0.24 s	<0.01 s

und somit eine Gesamtlaufzeit der Rechnung (ohne Einlesen der ASCII-Ausgangsdaten) von 0.29 Sekunden. In den folgenden Abbildungen sind Maximalgeschwindigkeit, korrigierte Maximalgeschwindigkeit, v(s)-Verlauf, t(s)-Verlauf, äußere Kräfte, Antriebskraft und zulässige Bremsbeschleunigung dargestellt.

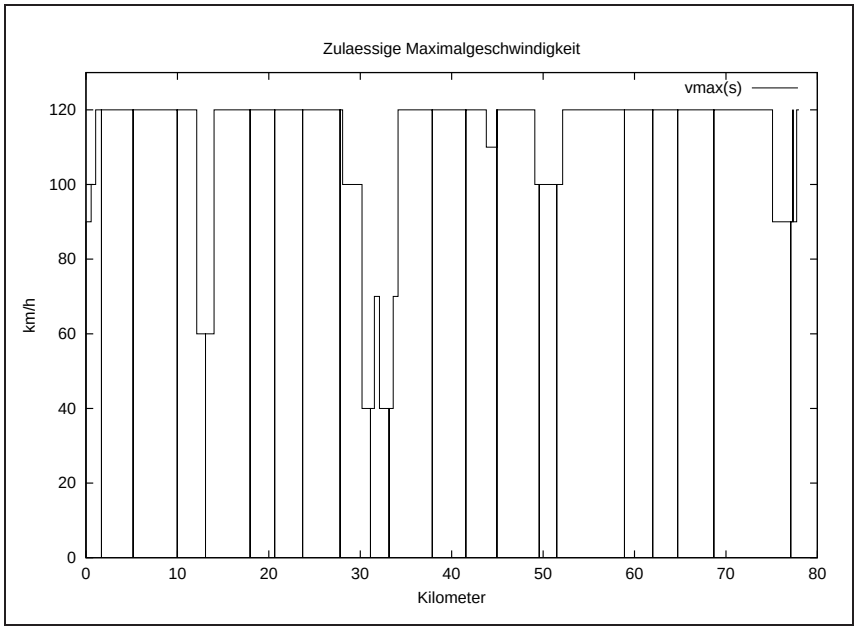

Abbildung 5.7: Beispiel 22127-1013: Zulässige Maximalgeschwindigkeit

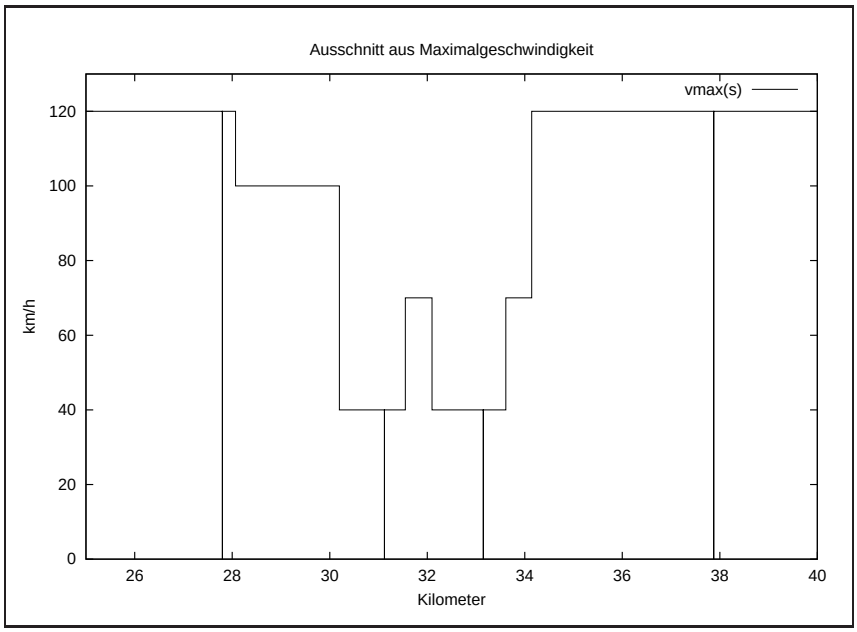

Abbildung 5.8: Beispiel 22127-1013: Ausschnitt aus der zulässigen Maximalgeschwindigkeit

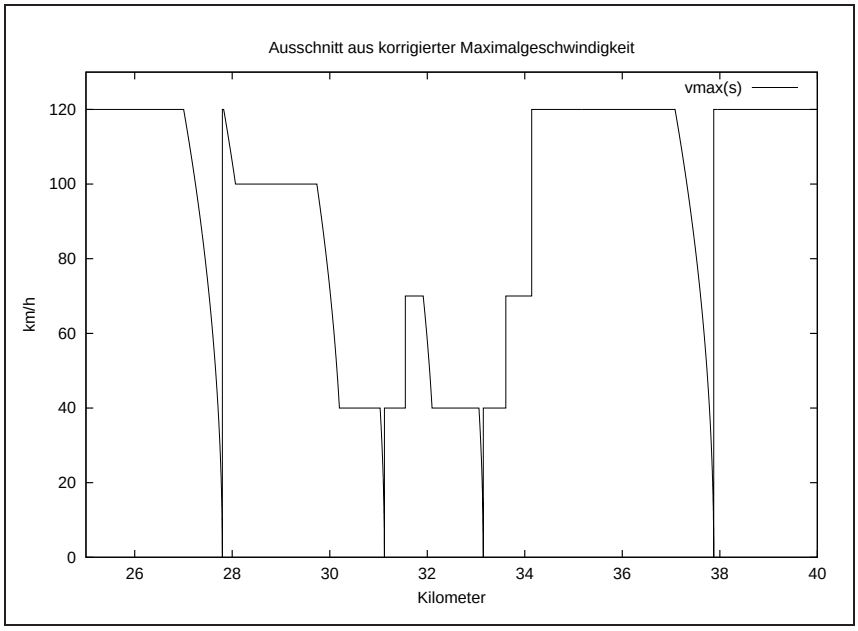

Abbildung 5.9: Beispiel 22127-1013: Korrigierte Maximalgeschwindigkeit

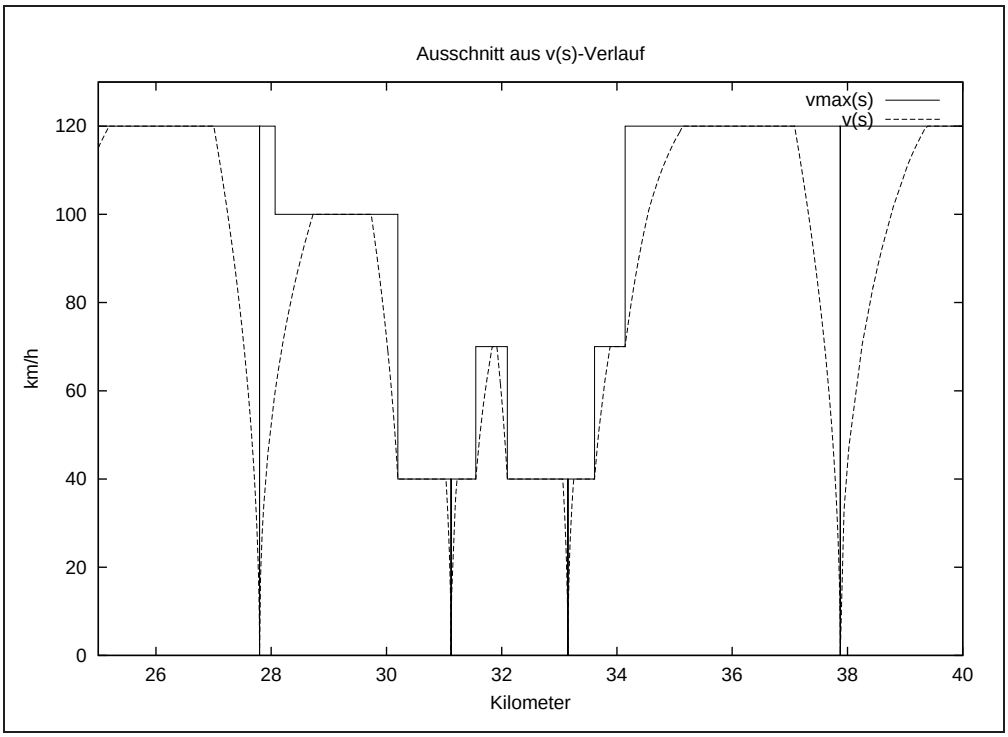

Abbildung 5.10: Beispiel 22127-1013: Geschwindigkeits-Weg-Verlauf

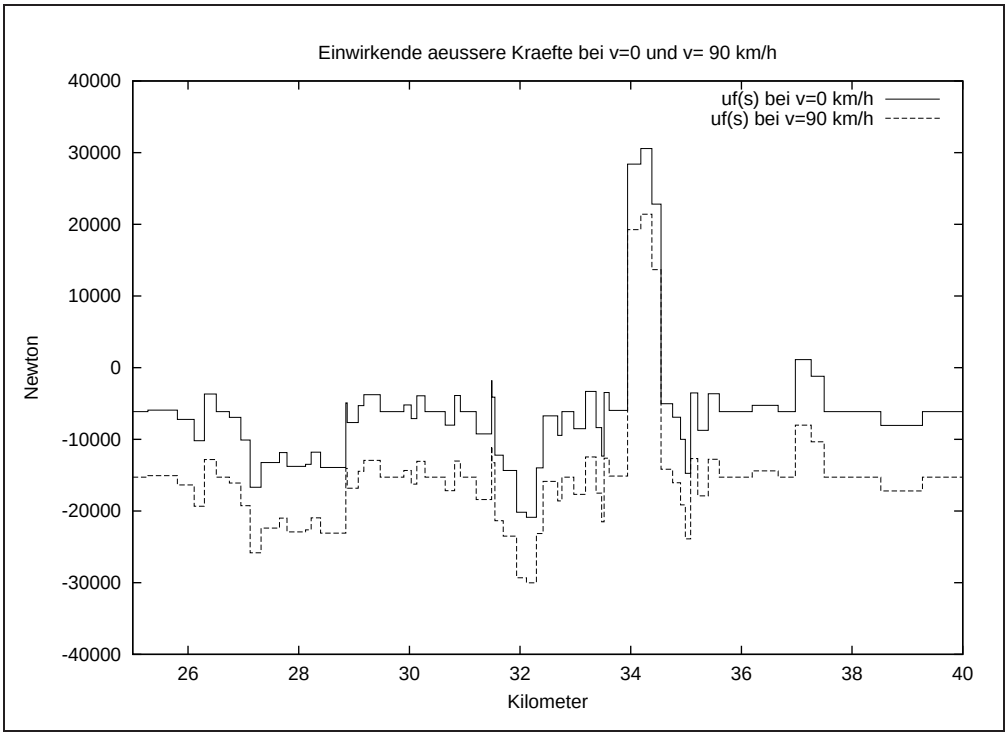

Abbildung 5.11: Beispiel 22127-1013: Äußere Kräfte

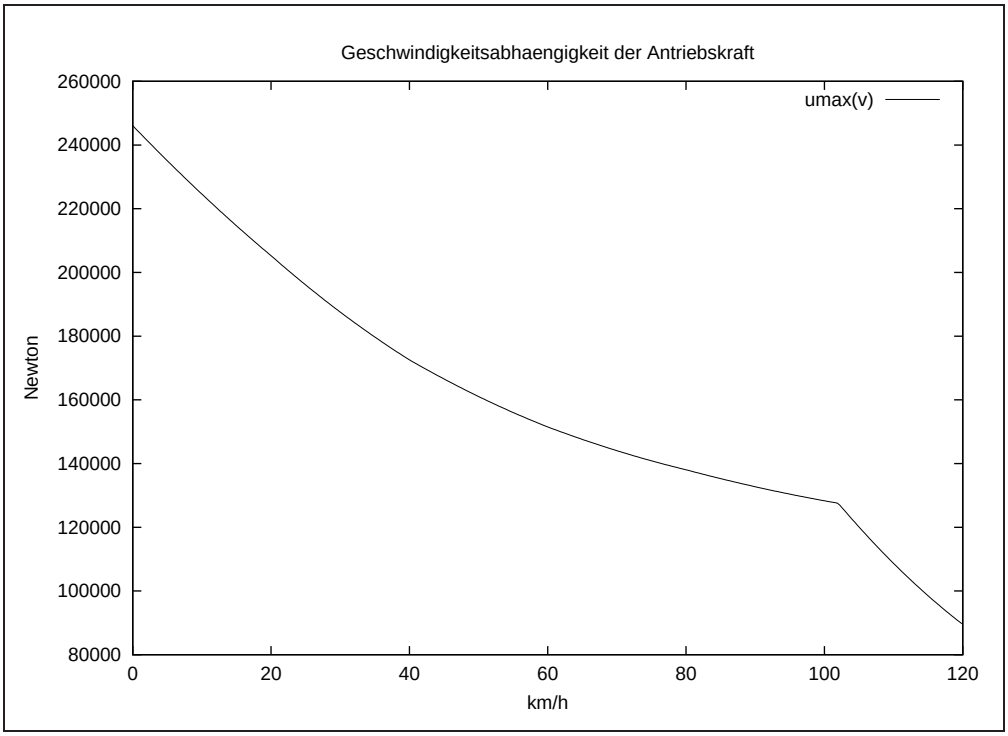

Abbildung 5.12: Beispiel 22127-1013: Maximale Antriebskraft

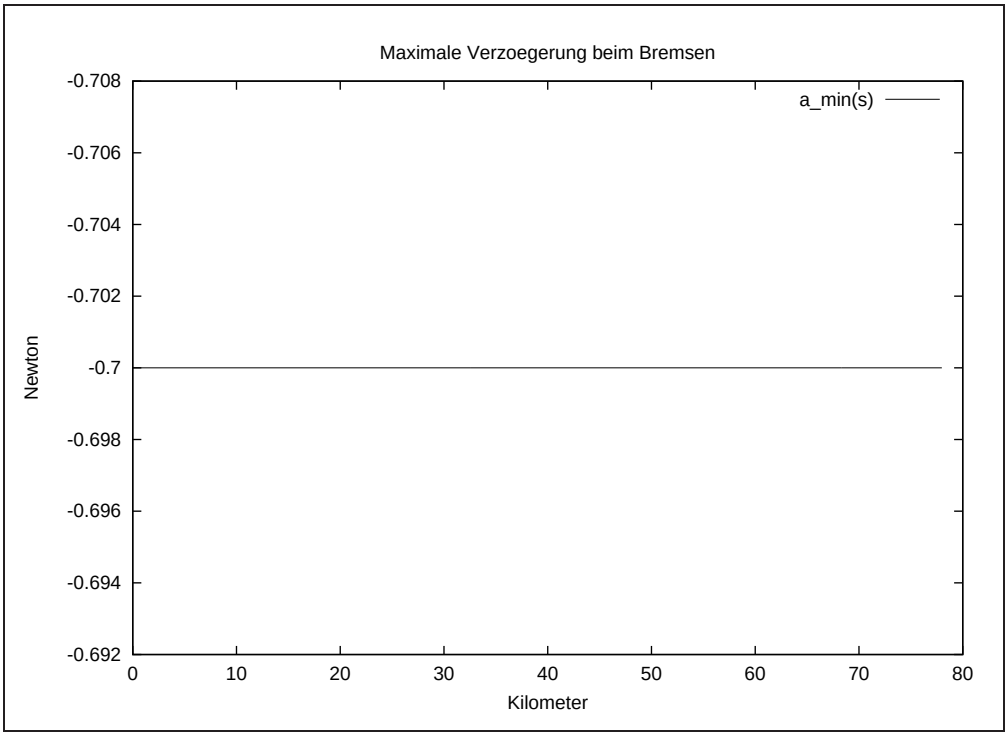

Abbildung 5.13: Beispiel 22127-1013: Bremsverzögerung

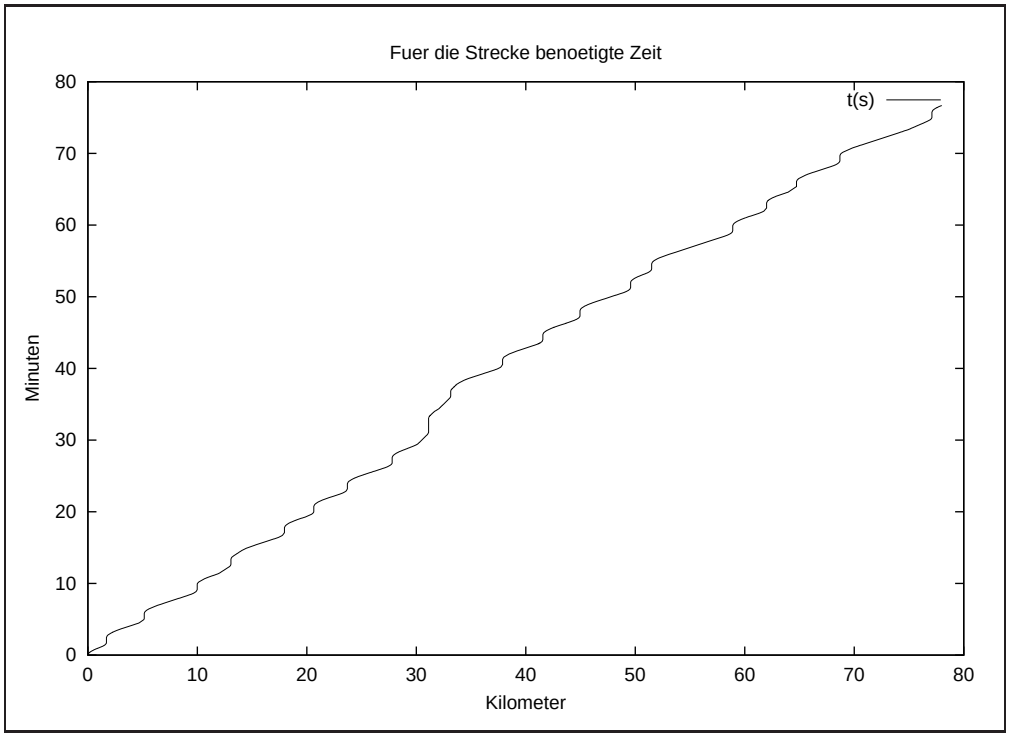

Abbildung 5.14: Beispiel 22127-1013: Zeit-Weg-Verlauf

Kapitel 6

Übertragung des Verfahrens auf den Fall mehrerer Züge auf einem Netz

6.1 Grundlagen

In den bisherigen Betrachtungen wurde von einem einzelnen Zug ausgegangen, der ungehindert fahren durfte.
Im weiteren betrachten wir mehrere Züge, die auf einem Schienennetz fahren.
Für jeden Zug sei sein Laufweg durch das Netz, die zugehörigen Streckenparameter und die Startzeit der Fahrt a priori festgelegt.

Definition 7 (Laufweg) *Das Intervall* $L^i := [s_a^i, s_e^i] \subset \mathbf{R}$ *wird als Laufweg des i-ten Zuges bezeichnet.*

Auf dem Laufweg seien die folgenden stückweise konstanten Funktionen definiert:

- Die zulässige Höchstgeschwindigkeit $v_{max}^i : L^i \to \mathbf{R}_0^+$.
- Die Neigung der Strecke $\omega^i : L^i \to \mathbf{R}$.
- Die maximal zulässige Bremsverzögerung $a_{min}^i : L^i \to \mathbf{R}^-$.

Der Zustand des Zuges sei wie bisher durch den Vektor

$$x^i(t) := \begin{bmatrix} x_1^i(t) \\ x_2^i(t) \end{bmatrix} \tag{6.1}$$

bezeichnet.
Dabei seien $x_1^i(t)$ der Ort und $x_2^i(t)$ die Geschwindigkeit des i-ten Zuges zum Zeitpunkt t.
Neu ist jedoch die Indizierung mit i, die sich auf die Nummer des Zuges bezieht, da jetzt mehr als nur ein Zug zu betrachten ist.
Dabei seien die Züge von 1 bis zu der Anzahl n_z durchnumeriert.
Im weiteren erarbeiten wir ein Modell des Sicherungskonzeptes der Deutschen Bahn, um Kollisionen von Zügen zu verhindern.

6.2 Das Sicherungskonzept

Im Falle von mehreren Zügen, die auf einem zweidimensionalen Netz fahren, kommt es zu einer gegenseitigen Beeinflussung.
Um Kollisionen der Züge zu verhindern, ist eine Sicherung der Gleisabschnitte nötig.
Grundlage dieses Sicherungskonzeptes ist eine Einteilung der Laufwege der Züge in einzelne Abschnitte, die im weiteren Sicherungsblöcke genannt werden.

Definition 8 (Sicherungsblock) *Ein Sicherungsblock sei ein abgeschlossenes Intervall* $S_j \subset L^i$.

Bemerkung 3 *Die Sicherungsblöcke sind von 1 bis zu der Anzahl der vorhandenen Sicherungsblöcke durchnumeriert.*
Dadurch gehört ein Sicherungsblock zu genau einem Zug.
Wenn also z. B. zwei Züge auf der gleichen Strecke (mit zeitlichem Versatz) fahren, dann haben trotzdem ihre Sicherungsblöcke verschiedene Nummern. Im weiteren sei $m_s \in \mathbf{N}$ *die Anzahl der Sicherungsblöcke.*

Die praktische Realisierung der Sicherungsblöcke sieht so aus, daß ein Sicherungsblock immer von einem Vorsignal bis zu einem Hauptsignal reicht. Eine einfache Grundregel ist die, daß ein Zug immer mindestens ein rotes Signal hinter sich haben muß. Die Sicherung wird dabei durch Bremskurven modelliert.

Definition 9 (Bremskurve) *Eine Bremskurve sei eine, auf einem abgeschlossenen Intervall* $[s_1, s_2]$ *definierte, monoton fallende Funktion*

$$b : [s_1, s_2] \subset \mathbf{R} \to \mathbf{R}_0^+ \tag{6.2}$$

mit der Eigenschaft

$$b(s_2) = 0 \tag{6.3}$$

In jedem Sicherungsblock ist eine Bremskurve

$$b_j : S_j \to \mathbf{R}_0^+ \tag{6.4}$$

als Geschwindigkeitsrestriktion sowie eine, von den Zuständen aller Züge abhängige, Indikatorfunktion

$$\chi_j : \mathbf{R}^{n_z} \to \{0, 1, 2, \cdots\} \tag{6.5}$$

$$\chi_j(x^1 \cdots x^{n_z}) := \begin{cases} 0 & \text{Hauptsignal zeigt grün} \\ > 0 & \text{Hauptsignal zeigt rot} \end{cases} \tag{6.6}$$

derart festgelegt, daß der Zug im Falle $\chi_j > 0$ bei Einhaltung dieser Restriktion am Ende des Blockes zum Halten kommt.
Diese Indikatorfunktionen können ihre Werte genau dann ändern, wenn irgendein Zug in einen Sicherungsblock einfährt oder ihn verläßt.

Dabei bewirkt das Einfahren eines Zuges in einen Sicherungsblock, daß in den zugehörigen anderen Sicherungsblöcken χ seinen Wert um 1 erhöht, d. h. Hauptsignal geht auf Rot bzw. bleibt auf Rot.
Beim Verlassen eines Sicherungsblockes hingegen kann χ in anderen Sicherungsblöcken seinen Wert um 1 verringern, falls $\chi > 0$ war.
Dadurch, daß die Sicherungsblöcke von Vorsignal bis zum übernächsten Hauptsignal reichen, befindet sich ein Zug immer in mindestends einem und höchstens zwei Sicherungsblöcken. (Siehe dazu Bild 6.1)

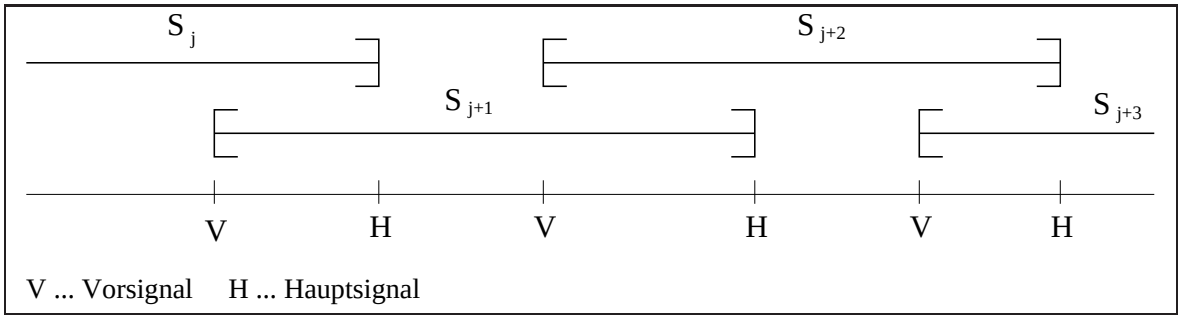

Abbildung 6.1: Verteilung der Sicherungsblöcke eines Zuges über seinem Laufweg

Um im weiteren eine feste Anzahl von Sicherungsblöcken verwenden zu können, definieren wir uns einen Dummyblock für die Fälle, wo sich der Zug nur in einem Sicherungsblock befindet.

Definition 10 (Dummyblock) *Der Dummyblock sei der Sicherungsblock S_0 für alle Züge mit der Eigenschaft, daß*

$$\chi_0(x^1(t) \cdots x^{n_z}(t)) = 0 \quad \forall t \quad . \tag{6.7}$$

Bemerkung 4 *Dies bewirkt, daß in diesem Block niemals eine Bremskurve aktiv ist.*

Dieser Dummyblock wird in die „Lücken“ eingesetzt (siehe Bild 6.2) und man erreicht, daß sich der Zug immer in genau zwei Sicherungsblöcken befindet.
Diese beiden Sicherungsblöcke seien im weiteren lokal mit Eins und Zwei numeriert und durch die folgenden Funktionen

$$\sigma^i : \{1, 2\} \times \mathbf{R} \to \{0, 1, \cdots m_s\} \tag{6.8}$$

erfolgt die Zuordnung der lokalen Nummern der Sicherungsblöcke auf die globale Nummer in Abhängigkeit vom momentanen Ort des Zuges i .
Es gibt jetzt also für jeden Zug genau 3 Geschwindigkeitsrestriktionen, nämlich die durch die Strecke vorgegebene zulässige Höchstgeschwindigkeit $v^i_{max}(s)$ sowie die, durch die beiden Bremskurven vorgegebenen folgenden Restriktionen

$$v^i_{max1}(t, s) := \begin{cases} b_{\sigma(1,s)} & \text{falls} \quad \chi_j(x^1(t) \cdots x^{n_z}(t)) > 0 \\ +\infty & \text{sonst} \end{cases} \tag{6.9}$$

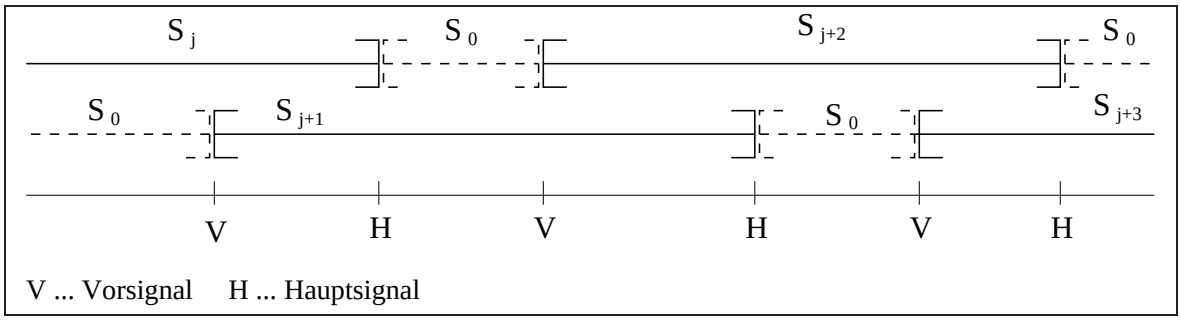

Abbildung 6.2: Einsetzen der Dummyblöcke

$$v^i_{max2}(t,s) := \begin{cases} b_{\sigma(2,s)} & \text{falls} \quad \chi_j(x^1(t)\cdots x^{n_z}(t)) > 0 \\ +\infty & \text{sonst} \end{cases} \tag{6.10}$$

Als relevante Restriktion $v^i_{rel}(t,s)$ tritt damit das Minimum dieser drei Restriktionen

$$v^i_{rel}(t,s) := \min\{v^i_{max}(s), v^i_{max1}(t,s), v^i_{max2}(t,s)\} \tag{6.11}$$

im weiteren als die einzuhaltende Geschwindigkeitsrestriktion für alle Züge $i, i = 1\cdots n_z$ auf.

6.3 Modellierung der Indikatorfunktionen χ_j

Die Indikatorfunktionen $\chi_j(x^1(t)\cdots x^{n_z}(t))$ stellen die Anzahlen der Sperrungen im Sicherungsblock j dar.
Ist $\chi_j(x^1(t)\cdots x^{n_z}(t)) = 0$ dann ist der Block frei und darf verlassen werden.
Wenn $\chi_j(x^1(t)\cdots x^{n_z}(t)) = k$ mit $k \neq 0$, dann liegen genau k Sperrungen auf dem Sicherungsblock.
Im Allgemeinen wird k gleich Eins sein, wenn z. B. zwei Züge hintereinander fahren, es ist aber durchaus möglich, daß k größer Eins ist.
Das tritt vor allem dann auf, wenn an bestimmten Stellen ein Zug solange warten muss, bis mehrere andere Züge diese Stelle passiert haben.
Wir hatten bereits gezeigt, daß sich die Werte von χ nur dann ändern, wenn irgendein Zug einen Sicherungsblock verläßt oder in einen Sicherungsblock einfährt. Diese Änderungen waren dadurch gekennzeichent, daß beim Einfahren Anzahlen von Sperrungen erhöht und beim Verlassen von Sicherungsblöcken Anzahlen von Sperrungen gesenkt werden können. Außerdem wird per Definition jeder Sicherungsblock genau einem Zug zugeordnet.
Dadurch ist es möglich, die Änderungen der Indikatorfunktion χ einzig und allein über eine Anfangsbelegung und eine Verknüpfung der Sicherungsblöcke über Punkt-Menge-Abbildungen zu ermitteln.

Definition 11 (Punkt-Menge-Abbildung) *Sei eine beliebige Menge M gegeben und bezeichne $\mathbf{P}(M)$ die Potenzmenge, (d.h. die Menge aller Teilmengen) von M.*

Eine Zuordnung $F : M \to \mathbf{P}(M)$ heißt Punkt-Menge-Abbildung (PMA), wenn jedem $m \in M$ eine Teilmenge $F_m \subset M$ zugeordnet ist. F ist dabei eine Abbildung von M in $\mathbf{P}(M)$.

Sei der Vektor $\beta \in \mathbf{N}_0^{m_s}$ der Vektor mit den Anzahlen von Sperrungen in den einzelnen Sicherungsblöcken.
Zum Startzeitpunkt t_0 wird dieser Vektor initialisiert mit

$$\beta_j := \chi_j(x^1(t_0) \cdots x^{n_z}(t_0)) \qquad j = 1 \cdots m_s \tag{6.12}$$

Desweiteren seien die folgenden Punkt-Menge-Abbildungen

$$\text{sperr} : \{1 \cdots m_s\} \to \mathbf{P}(\{1 \cdots m_s\}) \tag{6.13}$$

$$\text{entsperr} : \{1 \cdots m_s\} \to \mathbf{P}(\{1 \cdots m_s\}) \tag{6.14}$$

gegeben.
Diese beiden Punkt-Menge-Abbildungen sind durch die Streckenkonzeption im Preprocessing festgelegt und tabellarisch erfassbar.
Als ein Beispiel siehe Tabelle 6.1.

j	`sperr`(j)	`entsperr`(j)
1	$\{2, 6, 7\}$	$\{3, 4\}$
2	$\emptyset$	$\{1, 23, 12\}$
$\cdots$	$\cdots$	$\cdots$
m_s	$\{8, 2\}$	$\{9\}$

Tabelle 6.1: Beispiel einer Tabellierung der Punkt-Menge-Abbildungen

Mit diesen beiden PMA lassen sich nun beim Einfahren eines Zuges in den Sicherungsblock $\hat{j}$ die Sperrungen anderer Blöcke ermitteln als

$$\beta_i := \beta_i + 1 \qquad \forall i \in \text{sperr}(\hat{j}) \quad . \tag{6.15}$$

Analog erhält man die Entsperrungen beim Verlassen von Blöcken, unter Beachtung, daß $\beta_i \geq 0 \quad i = 1 \cdots m_s$ zu sichern ist als

$$\beta_i := \beta_i - 1 \qquad \forall i \in \text{entsperr}(\hat{j}) \quad \text{mit} \quad \beta_i > 0 \quad . \tag{6.16}$$

6.4 Beispiele verschiedener Interaktionen zweier Züge

In diesem Abschnitt wollen wir die Herleitung der PMA `sperr(j)` und `entsperr(j)` an verschiedenen Beispielen der Interaktion zweier Züge auf einem Netz herleiten.
Dabei ist es vorteilhaft, für die Konzeption zwischen sicherungstechnisch bedingten Sperrungen und Sperrungen für die Zugfolgeregelung zu unterscheiden.
In der Modellierung lassen sich diese beiden Teile jedoch über passende Initialbelegungen von β miteinander vereinigen und müssen in der Simulation nicht unterschieden werden.

6.4.1 Hintereinanderfahren zweier Züge

Fahren zwei Züge auf der gleichen Strecke hintereinander, so ist sicherzustellen, daß der hintere Zug nicht auf den vorderen Zug auffährt. Für den vorderen Zug hingegen gibt es keine Beschränkungen. Zur Entwicklung der Sicherung gibt es zwei grundsätzliche Möglichkeiten.
Zum einen kann der vordere Zug immer die Sicherungsblöcke hinter sich sperren und nach Passieren entsprechend entsperren, zum anderen kann man die Sicherung auch so gestalten, daß mit der Initialbelegung für den hinteren Zug die gesamte Strecke gesperrt wird und die Sicherungsblöcke durch den vorderen Zug freigegeben werden. Dieses Konzept wollen wir zuerst anwenden.

Sperrung über Initialbelegung

Wir haben also m_s Sicherungsblöcke insgesamt, davon gehören die Sicherungsblöcke $1 \cdots m_v$ zum vorderen Zug und die restlichen Sicherungsblöcke $m_v + 1 \cdots m_s$ zum hinteren Zug. Wir gehen weiterhin in diesem Beispiel davon aus, daß die örtliche Verteilung der Sicherungsblöcke auf der Strecke für beide Züge identisch ist, und die Anzahl m_s sei gerade. Dann gilt, daß $m_v = \frac{1}{2} m_s$ und die Sicherungsblöcke S_j und S_{j+m_v} $j = 1 \cdots m_v$ liegen auf gleichen Abschnitten der zu fahrenden Strecke. Siehe dazu auch Bild 6.3.
Damit ergibt sich als Initialbelegung für den Vektor β

$$\beta_j = \begin{cases} 0 & j = 1 \cdots m_v \\ 1 & j = m_v + 1 \cdots m_s \end{cases} . \tag{6.17}$$

Betrachten wir den Fall, daß sich der vordere Zug in S_v befindet und diesen nun an seinem rechten Ende verläßt. Dann darf der hintere Zug den Block S_{v-1} komplett benutzen, d. h. die Bremskurve in dem mit Block S_{v-2} korrespondierenden Block vom hinteren Zug kann abgeschaltet werden. Dies ist (mit den Bezeichnungen vom Bild 6.3) der Block S_{h-2}. Die Indizes h und v lassen sich durch $h = v + m_v$ ineinander überführen. Somit entsperrt der vordere Zug beim Verlassen des Blockes S_v den Block S_{v+m_v-2} und die PMA `sperr(j)` und `entsperr(j)` haben die folgende Gestalt:

$$\texttt{sperr(j)} = \emptyset \qquad j = 1 \cdots m_s \tag{6.18}$$

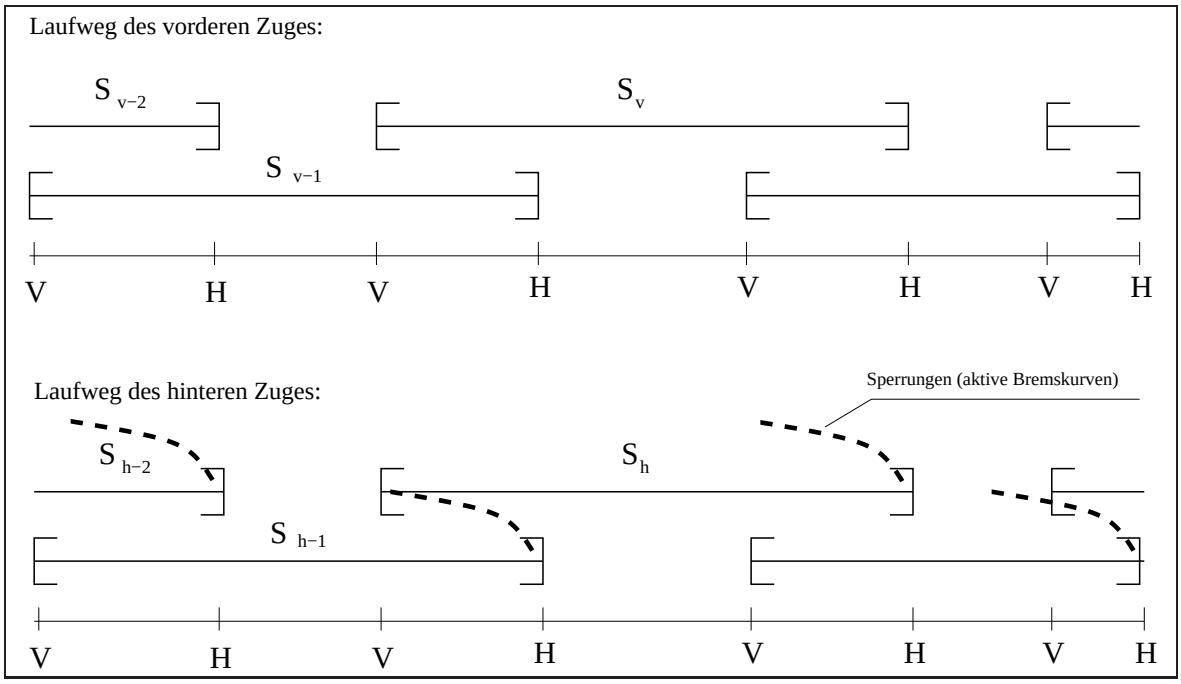

Abbildung 6.3: Zwei Züge hintereinander auf der selben Strecke

$$\texttt{entsperr(j)} = \begin{cases} \emptyset & j = 1 \cdots 2 \\ \{v + m_v - 2\} & j = 3 \cdots m_v \\ \emptyset & j = m_v + 1 \cdots m_s \end{cases} \tag{6.19}$$

Sperrung durch den vorderen Zug

Die zweite Möglichkeit ist die Konzeption, daß ein Zug die hinter sich liegenden, zu sichernden Blöcke selbst sperrt und nach Passieren der entprechenden anderen Blöcke wieder entsperrt.
Wir beginnen also mit einer freien Initialbelegung, d.h. $\beta_j = 0 \; \forall j$.
Betrachten wir wieder den vorderen Zug, wenn er sich im Block S_v befindet. Dann ist dieser Block für ihn exklusiv und der hintere Zug darf den Block nicht befahren. Dies wird gewährleistet, wenn die Bremskurven am Ende der zu S_{v-1} und S_{v-2} korrespondierenden Blöcke aktiviert werden. Dabei reicht es aus, die Bremskurve in dem auf der Strecke weiter zurückliegenden Block zu aktivieren, da dann indirekt der andere Block mit gesperrt ist. Die Entsperrung erfolgt wieder analog zu ersten Konzept und es ergeben sich die PMA zu

$$\texttt{sperr(j)} = \begin{cases} \emptyset & j = 1 \cdots 2 \\ \{v + m_v - 2\} & j = 3 \cdots m_v \\ \emptyset & j = m_v + 1 \cdots m_s \end{cases} \tag{6.20}$$

$$\texttt{entsperr(j)} = \begin{cases} \emptyset & j = 1 \cdots 2 \\ \{v + m_v - 2\} & j = 3 \cdots m_v \\ \emptyset & j = m_v + 1 \cdots m_s \end{cases} \tag{6.21}$$

6.4.2 Einmündung

Beim Befahren einer Weiche ist grundsätzlich sicherzustellen, daß es nicht zu einer Kollision der beiden Züge kommen kann.
Außerdem kann man aber durch entsprechende Initialbelegungen und Entsperrungen erreichen, daß die Züge die Weiche nur in einer bestimmten Reihenfolge passieren können. Dies ist durchaus sinnvoll, wenn man gewährleisten will, daß der schnellere Zug die Weiche zuerst passiert um dann auf der folgenden (vielleicht sehr langen) eingleisigen Strecke vorn zu fahren. Anderenfalls könnte es nämlich passieren, daß der langsame Zug nach vorn kommt und damit den anderen auf der ganzen Strecke ausbremst.
Wir haben also (wie im Bild 6.4) eine linken und einen rechten Zug, die die Weiche passieren wollen.

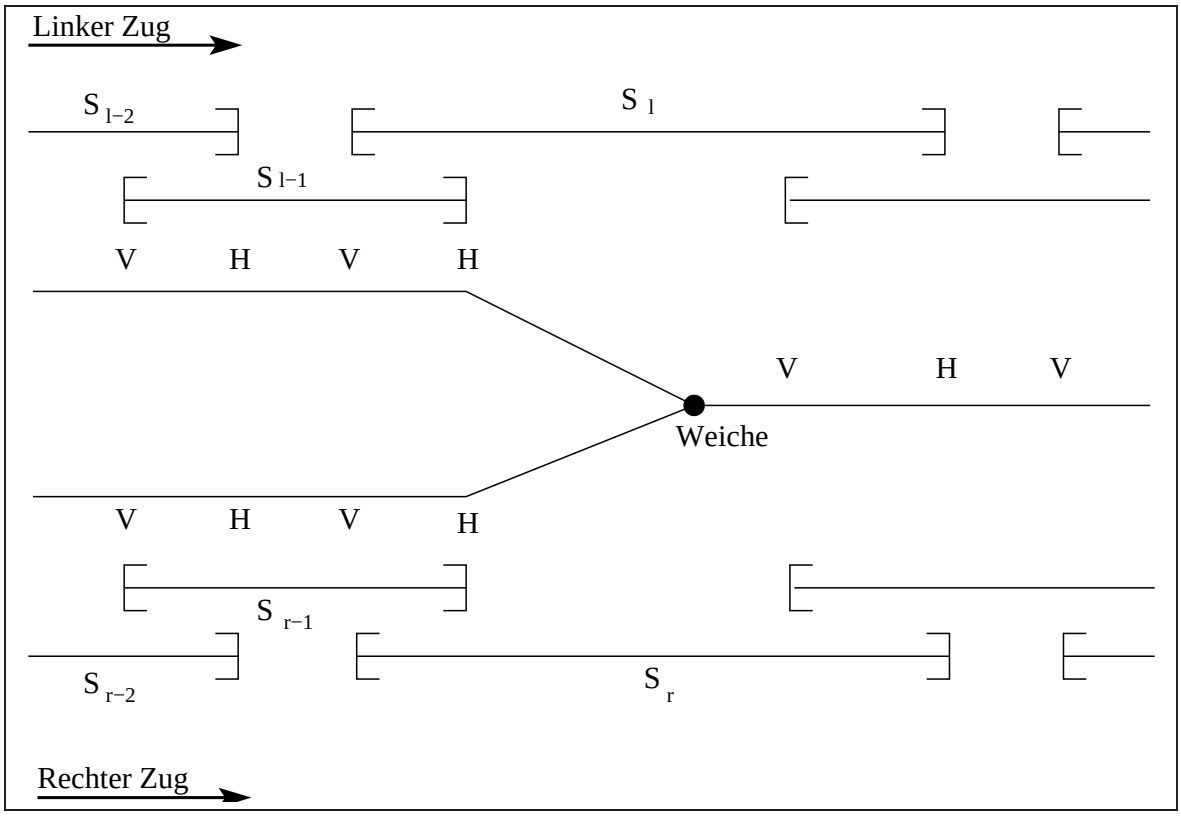

Abbildung 6.4: Zwei Züge passieren eine Weiche

Die interessanten Sicherungsblöcke des linken Zuges seien mit $l-2$, $l-1$, l, $l+1\cdots$ indiziert, die des rechten Zuges mit $r-2$, $r-1$, r, $r+1\cdots$.

Die beiden, über die Weiche gehenden Sicherungsblöcke sind S_l und S_r, man hat verschiedene Möglichkeiten die Weiche zu sichern. Soll nur die Weiche gesichert werden, so können das die beiden letzten Hauptsignale vor der Weiche erreichen. Beim Einfahren eines Zuges in S_l ist dann die Bremskurve in S_{r-1} zu aktivieren und umgekehrt beim Einfahren in S_r die Bremskurve in S_{l-1}.
Will man nicht nur die Weiche sichern, sondern auch erreichen, daß sich nur entweder in S_l oder in S_r ein Zug befinden darf, so ist die gleiche Vorgehensweise noch auf ein weiter zurückliegendes korrespondierendes Paar anzuwenden.

6.5 Einfluß des Sicherungskonzeptes auf die Bewegungs-Differentialgleichung

Jeder Zug für sich genommen bewegt sich nach dem gleichen Bewegungsgesetz, wie es im Falle eines Zuges in Gleichung 1.6 hergeleitet wurde.
Für die zeitoptimale Steuerung des Zuges über seinen Laufweg ohne Geschwindigkeitsvorgabe im Endzustand, und ohne Beachtung von Zustandsbeschränkungen gilt also wiederum

$$\begin{cases} (t_{end} - t_0) \rightarrow \min \\ \dot{x^i}(t) = Ax^i(t) + B^i[u^i_f(s,v) + u^i_v(t)] \\ x^i(t_0) = \begin{bmatrix} s^i_0 \\ v^i_0 \end{bmatrix} \\ x^i(t_{end}) = \begin{bmatrix} s^i_{end} \\ \cdot \end{bmatrix} \\ u^i_v(t) \in [u^i_{min}, u^i_{max}(x^i_2(t))] \end{cases} \tag{6.22}$$

wobei die Matrix A für alle Züge gleich ist, jedoch die Matrix B für jeden Zug eine andere, da sie die Masse des Zuges enthält.

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{6.23}$$

$$B^i = \begin{bmatrix} 0 \\ \frac{1}{m^i} \end{bmatrix} \qquad i = 1 \cdots n_z \tag{6.24}$$

Die einzige Zustandsbeschränkung im Falle eines Zuges war die Geschwindigkeitsrestriktion

$$x_2(x_1(t)) \in [v_{min}(x_1(t)), v_{max}(x_1(t))] \quad \forall t \quad . \tag{6.25}$$

Diese streckenbezogene Restriktion überträgt sich analog auf den Fall mehrerer Züge

$$x^i_2(x^i_1(t)) \in [v^i_{min}(x^i_1(t)), v^i_{max}(x^i_1(t))] \quad i = 1 \cdots n_z \quad \forall t \quad . \tag{6.26}$$

Jedoch kommen weitere Zustandsbeschränkungen hinzu, die die Züge miteinander verkoppeln.
Wir hatten im Abschnitt über die Grundlagen des Sicherungskonzeptes die in den Sicherungsblöcken vorgegebenen Bremskurven $b_j(s)$ (vgl. 6.4) und die Indikatorfunktionen

$\chi_j(x^1 \dots x^{n_z})$ (vgl. 6.5), die in Abhängigkeit vom Ort der einzelnen Züge die Bremskurven aktiviert, hergeleitet und letztendlich die relevanten Geschwindigkeitsrestriktion $v^i_{rel}(t,s)$ (vgl. 6.11) erhalten.
Da die Indikatorfunktionen $\chi_j(\cdots)$ von den momentanen Orten aller Züge (die zeitabhängig sind) abhängt, haben wir jetzt eine orts- und zeitabhängige Restriktion an den Zustand. Da die Zeitabhängigkeit aber indirekt über die Orte der Züge gegeben ist, kann man nicht mehr jeden Zug für sich betrachten, sondern man erhält ein gekoppeltes System für alle Züge im Netz.

$$\begin{cases} (t_{end} - t_0) \rightarrow \min \\ \dot{x^i}(t) = Ax^i(t) + B^i[u^i_f(s,v) + u^i_v(t)] \\ x^i(t_0) = \begin{bmatrix} s^i_0 \\ v^i_0 \end{bmatrix} \\ x^i(t_{end}) = \begin{bmatrix} s^i_{end} \\ \cdot \end{bmatrix} \\ u^i_v(t) \in [u^i_{min}, u^i_{max}(x^i_2(t))] \\ x^i_2(t) \in [v^i_{min}(x^i_1(t)), v^i_{rel}(t, x^i_1(t))] \end{cases} \tag{6.27}$$

Diese neuen Zustandsrestriktionen bedingen die Wahl des variablen Teiles der Steuerung $u^i(t)$.
Da nach wie vor die zeitoptimale Steuerung gesucht ist, gibt es nur 3 qualitativ verschiedene Schaltzustände der Steuerung.
In [1] wurde für den zeitoptimalen Fall ohne Zustandsbeschränkung gezeigt, daß die optimale Steuerung nur ihre beiden Extremalwerte (Minimum und Maximum) annimmt.
Dies wird auch bang-bang-Steurung genannt.
In unserem Fall mit Zustandsrestriktionen gibt es noch einen dritten Fall, nämlich den, daß die Zustandsrestriktion erreicht wird, d.h. es wird auf die Restriktion aufgesprungen.
Dabei seien die Teile der Restriktion betrachtet, bei denen es sich nicht um Bremskurven handelt, denn in diesen Bereichen geht die Steuerung auf ihren Minimalwert, dies ist also einer der beiden Fälle der normalen bang-bang-Steuerung.
Es bleiben also nur diese Phasen für den dritten Fall übrig, in denen auf eine konstante zulässige Höchstgeschwindigkeit aufgesprungen wird.
In diesen Phasen muß der variable Teil der Steuerung so geschaltet werden, daß er die äußeren Kräfte kompensiert. Für die gesamte Steuerung $u(t)$, die ja aus dem variablen und festen Anteil besteht bedeutet dies, daß sie zu Null wird.
Dadurch ist die Bewegung in Bereichen, in denen auf die Restriktion aufgesprungen wurde, durch diese Restriktion selbst vorgegeben.
Es gilt dann für die Entwicklung des Zustandes

$$\dot{x}^i_1(t) = x^i_2(t) \tag{6.28}$$
$$x^i_2(t) = v^i_{rel}(t, x^i_1(t)) \tag{6.29}$$

für alle Zeitpunkte t, in denen auf der Zustandsrestriktion gefahren wird.
Es läßt sich somit die Entwicklung des Zustandes bei vorgegebenen Anfangswerten für alle Zeiten t bestimmen.

Kapitel 7

Das Simulationsverfahren für mehrere Züge

Um die kürzesten Fahrzeiten mehrerer Züge mit dem hergeleiteten Sicherungskonzept ermitteln zu können folgt die Beschreibung des Simulationskonzeptes.

7.1 Grundidee

Die Simulation erfolgt in vorgegebenen Zeitschritten, startend zur Simulationszeit $t = 0$, bis alle Züge angekommen sind.
Analog wie bei der Variante für einen Zug werden die Bremskurven vor dem Start der Simulation ermittelt, so daß sich das Verfahren in folgende Schritte gliedern läßt:

1. Anlegen der nötigen Datenstrukturen und Einlesen der Daten.
2. Eventuell Überprüfen der Eingangsdaten auf Korrektgestelltheit.
3. Ausglätten der Abwärtssprünge der streckenbezogenen Maximalgeschwindigkeit.
4. Berechnen der Bremskurven des Sicherungskonzeptes
5. Start und Ablauf der Simulation
6. Postprocessing

7.2 Datenstukturen

Die Verwaltung der Züge erfolgt über einen Vektor, in dem die einzelnen Züge als Objekte Zugfahrt abgelegt sind.
Jede Zugfahrt ihrerseits umfaßt sämtliche Daten zu Fahrzeug und Strecke, sowie die nötigen Methoden zur Simulation.
Streckendaten sind als doppelt verkettete Liste von einzelnen Wegabschnitten in jeder Zugfahrt gespeichert.

Außerdem gibt es noch ein Objekt zur Verwaltung der Sicherungsblöcke und zur Realisierung der Methoden zum Sperren und Entsperren einzelner Sicherungsblöcke.
Die Eingabedaten werden aus ASCII-Files eingelesen, die Struktur dieser Files ist im Kapitel zur Handhabung des Programmes beschrieben.

7.3 Überprüfen der Daten

Da die Simulation nur mit korrekten Daten sinnvoll arbeiten kann, werden die Eingangsdaten zu den Sicherungsblöcken dahingehend überprüft, ob die Tabellierungen der Punkt-Menge-Abbildungen `sperr` und `entsperr` in sich korrekt sind.

7.4 Ausglätten der Abwärtssprünge der streckenbezogenen Maximalgeschwindigkeit

Um die Abwärtssprünge der Maximalgeschwindigkeit auszuglätten wird analog wie im Verfahren zur einfachen Zugfahrt die Wegabschnittsliste von hinten durchgegangen und entweder Absprungpunkte oder die zulässige höchste Geschwindigkeit am Anfang des Wegabschnittes ermittelt.
Wird ein Absprungpunkt gefunden, so wird der entspechende Wegabschnitt verkürzt und ein neuer Wegabschnitt eingefügt.
Dadurch wird erreicht, daß auch weiterhin Wegabschnitte mit konstantem und fallendem v_{max} unterschieden werden können.

7.5 Berechnen der Bremskurven des Sicherungskonzeptes

Die Berechnung der Bremskurven erfolgt ähnlich zur Ausglättung der Höchstgeschwindigkeit.
Jedem Wegabschnitt sind durch die Einführung des Dummyblockes genau 2 Sicherungsblöcke zugeordnet und über die beiden lokalen Abschnittsnummern `siblo1` und `siblo2` (die den Funktionen σ^i entsprechen) ansprechbar.
Zuerst werden die lokalen Nummern dahingehend geordnet, daß bei einem Wechsel des Wertes einer der beiden Variable auch von einem tatsächlichen Wechsel der Sicherungsblöcke ausgegangen werden kann.
Dies ist nämlich nicht von vornherein gegeben, denn im Datenfile könnten im (i+1)-ten Wegabschnitt die Nummern der Sicherungsblöcke im Vergleich zum i-ten Wegabschnitt einfach nur vertauscht worden sein.
Dies ist dann natürlich kein Wechsel der Sicherungsblöcke. Es ist auch möglich, daß nur ein Sicherungsblock tatsächlich wechselt, aber der andere im nächsten Abschnitt auf `siblo2` statt auf `siblo1` gespeichert wurde.
Folgender Algorithmus sichert, daß die Sicherungsblöcke korrekt gespeichert werden, und man bei einem Wechsel der Nummer auch von einem tatsächlichen Wechsel des Siche-

rungsblockes ausgehen kann. Dabei sei i ein Index zur Kennzeichnung eines Wegabschnittes, d.h. eines Listenelementes.

1. Starte am Anfang der Liste mit den Wegabschnitten d.h. setze i:=0
2. `nummer1`:=`siblo1`(i)
   `nummer2`:=`siblo2`(i)
3. Inkrementiere i
4. Falls `nummer1`=`siblo2`(i) dann tausche `siblo1`(i) und `siblo2`(i)
5. Falls `nummer2`=`siblo1`(i) dann tausche `siblo1`(i) und `siblo2`(i)
6. `nummer1`:=`siblo1`(i)
   `nummer2`:=`siblo2`(i)
7. Gehe zu 3 bis die Liste abgearbeitet ist.

Nun kann man die Liste wieder von hinten nach vorn durchgehen und die maximalen Startgeschwindigkeiten an den Anfangspunkten der Wegabschnitte ermitteln, die ein Anhalten am Ende des Sicherungsblockes gewährleisten.
Immer wenn ein Sicherungsblock endet, so ist an diesem Endpunkt die Maximalgeschwindigkeit gleich Null, daraus ermittelt man die Geschwindigkeit am Anfang des Wegabschnittes und nimmt diese als Ausgangspunkt für den vorherigen Wegabschnitt usw. bis der Sicherungsblock wieder wechselt. Dort ist dann die Maximalgeschwindigkeit wieder Null.
Auf diese Weise erhält man für alle Wegabschnitte die maximalen Startgeschwindigkeiten für die beiden Bremskurven und kann im weiteren in jedem Punkt des Wegabschnittes die Bremskurve $b_j(s)$ bestimmen.

7.6 Start und Ablauf der Simulation

Zur Initialisierung der Simulation wird auf einer Variable die Anzahl der noch fahrenden Züge gleich der Anzahl der Züge gesetzt.
Nun wird immer abwechselnd für jeden Zug ein Zeitschritt in der Simulation ausgeführt, bis alle Züge angekommen sind.
Dies erfolgt durch Aufruf der Simulationsmethode der Züge.
In dieser wird zuerst getestet, ob der Zug überhaupt noch fährt oder schon angekommen ist (in diesem Falle erfolgt sofort die Rückkehr zur Simulationsschleife).
Dann wird getestet, ob der Zug zur gegebenen Simulationszeit schon fahren darf oder noch warten muß, der Fall des Wartens tritt ein, falls die Simulationszeit noch früher als die Startzeit des Zuges ist, oder wenn er aufgrund eines Verkehrshaltes in einem Halteintervall seine Haltezeit einhalten muß.
In den anderen Fällen wird davon ausgegangen, daß der Zug fahren darf und es wird ein Zeitschritt berechnet.
Erfolgt dabei die Überschreitung einer Wegabschnittsgrenze so wird folgendes abgetestet:

- Wechselt ein Sicherungsblock ? Wenn ja dann teste ob der Block verlassen werden darf. Darf er verlassen werden, dann rufe die Methoden zum Entsperren und Sperren der zugehörigen anderen Sicherungsblöcke und verlasse ihn, darf er nicht verlassen werden, dann setze die eigene Position auf das Ende des Blockes und die Geschwindigkeit auf Null und warte.

- Ist das Ende der Strecke erreicht ? Wenn ja , dann interpoliere die Werte für Zeit und Geschwindigkeit am Zielpunkt, vermerke, daß der Zug angekommen ist und dekrementiere die Anzahl der noch fahrenden Züge.

- Ist in ein Halteintervall eingefahren worden ? Wenn ja, dann interpoliere die Ankunftszeit und setzte die Zeit bis zu der gewartet werden muß auf die Summe aus Ankunftszeit und Haltezeit.

Auf diese Weise erhält man eine Simulation der Bewegung aller Züge, es ist sinnvoll, während der Simulation Datenpunkte über die Zeit, den Ort und die Geschwindigkeit der Züge auszugeben, um im Postprocessing diese entsprechend verwerten zu können.

7.7 Postprocessing

Im Postprocessing erfolgt entsprechend den Vorgaben die Rückgabe ermittelter Werte zu Fahrzeiten der Züge, dies hängt vom entsprechenden Einsatz des Programmes ab.
In dieser Variante werden diese Daten einfach mit ausgegeben.

7.8 Programmierung

Für die Umsetzung des Simulationsverfahrens bietet sich wiederum die Programmiersprache C++ an, da sie durch das objektorientierte Konzept eine gute Darstellung ermöglicht. Die Einzelheiten der Programmteile, sowie die Ein- und Ausgabeparameter der Methoden sind in den Quelltexten zum Programm FZR2D dokumentiert.

Kapitel 8

Umgang mit dem Programm FZR2D

In diesem Kapitel soll das Programm zur Berechnung der Fahrzeiten, die verwendeten Eingabefiles und der Ablauf beschrieben werden.

8.1 Eingabedaten

Die Eingabe der Streckendaten, der Daten der Züge sowie der Sicherungskonzeption erfolgt über ASCII-Datenfiles, die im weiteren beschrieben werden. In den Eingabefiles sind Kommentarzeilen mit einem # in der ersten Spalte möglich, Leerzeilen sind nicht erlaubt.

8.1.1 Globales Startfile

Das globale Eingabefile ist beim Aufruf von fzr2d als Parameter anzugeben. In ihm werden die Anzahl der Züge, für jeden Zug der Name, sein Laufweg-File und sein Zugdaten-File festgelegt, und es wird der Name des Sicherungsblock-File angegeben.
Als Beispiel für die Bearbeitung von zwei Zügen das folgende File:

```
#
#  Beispiel eines Globalen Startfiles
#
#  Variante: Hintereinanderherfahren zweier Zuege
#
#  Alle Pfade in diesem File sind relativ zur
#  Position von fzr2d zu setzen
#
# Anzahl von Zuegen
 2
#
# Name , Laufweg-File und Zugdaten vom Zug 1
#
RB0815 Ich bin vorn
data/hintereinander1/laufweg-vorn.txt
```

```
data/hintereinander1/11522-ice2.txt
#
# Name, Laufweg-File und Zugdaten vom Zug 2
#
RB0816 Ich bin leider hinten
data/hintereinander1/laufweg-hinten.txt
data/hintereinander1/11522-ice2.txt
#
# Name des Files mit den Sicherungsbloecken
#
data/hintereinander1/siblo-test.txt
#
#  Ende des Datenfiles
#
```

8.1.2 Das Sicherungsblock-File

Im Sicherungsblock-File wird die Anzahl m_s von Sicherungsblöcken, die Initialbelegung des Vektors β sowie die Tabellierung der beiden Punkt-Menge-Abbildungen `sperr`(j) und `entsperr`(j) gespeichert.
Die Mengen `sperr`(j) werden dabei durch ein vorangestelltes großes S und die Mengen `entsperr`(j) durch ein großes E gekennzeichnet.
Auch wenn diese Mengen leer sind, sind S und E anzugeben.
Hier ein Beispiel eines solchen Files:

```
#
#  Beispiel eines Sicherungsblock-Files
#
#  Gesamtanzahl von Sicherungsbloecken
 10
#
#
# ******** Nun kommen die Daten der Sicherungsbloecke ***********
#
#  Fuer alle Bloecke aller Zuege wird wird pro Zeile festgelegt:
#    beta_j              ... Anfangsbelegung des Vektors beta
#                            0 = frei; sonst Anzahl n Sperrungen
#    card(sperr(j))      ... Kardinalitaet von sperr(j)
#    card(entsperr(j))   ... Kardinalitaet von entsperr(j)
#  ein S und die Elemente von sperr(j) und dann ein
#  E und die  Elemente von entsperr(j)
#
#  j          beta_j   card(sperr(j)) card(entsperr(j)) sperr(j) / entsperr(j)
# Bloecke des vorderen Zuges
   1            0          0            0                S   E
```

```
   2            0          0            1                S   E  6
   3            0          0            1                S   E  7
   4            0          0            1                S   E  8
   5            0          0            1                S   E  9
# Bloecke des hinteren Zuges
   6            1          0            0                S   E
   7            1          0            0                S   E
   8            1          0            0                S   E
   9            1          0            0                S   E
  10            0          0            0                S   E
#
#  Ende des Sicherungsblock-File
#
```

8.1.3 Das Laufweg-File

Im Laufwegfile eines Zuges wird die Startzeit und die Startgeschwindigkeit des Zuges, Anzahl von Wegabschnitten sowie Längen der einzelnen Streckenabschnitte, Neigung, zulässige Höchstgeschwindigkeit, zulässige maximale Bremsverzögerung, Haltezeit, die globalen Nummern der beiden lokalen Sicherungsblöcke sowie der Name des Abschnitts angegeben. Hier wieder ein Beispiel

```
#
#  Beispiel eines Laufweg-Files
#
#  Laufwegfile fuer Zug A, der faehrt vorn
#
#
# Startzeit des Zuges in Sekunden nach Null
 0.0
# Startgeschwindigkeit des Zuges in Kilometer pro Stunde
 0.0
#
#  Anzahl von Wegabschnitten
  9
# und nun die Daten der Abschnitte :
#
# laenge neigung vmax   amin    haltezeit Sich-bloecke       name
# [m]            [km/h] [m/s^2] [s]
  5000.0  0.0    100.0  -0.4      0.0     1    2      abschnitt_1
  3000.0  0.08   100.0  -0.4      0.0     0    2      abschnitt_2
  3500.0 -0.03   100.0  -0.4      0.0     2    3      abschnitt_3
     0.0  0.0      0.0  -0.4    600.0     2    3      bahnhof_xy
  1000.0  0.03    80.0  -0.4      0.0     0    3      abschnitt_4
   200.0  0.0     30.0  -0.4      0.0     0    3      abschnitt_5
```

```
  2000.0  0.03    80.0  -0.4      0.0     3    4      abschnitt_6
  2000.0 -0.01    60.0  -0.4      0.0     0    4      abschnitt_7
  1000.0  0.0    100.0  -0.4      0.0     4    5      abschnitt_8
# ENDE
```

8.1.4 Das Zugdaten-File

Im Zugdaten-File werden die fahrzeugrelevanten Daten festgelegt. Das ist zuerst die Masse des Zuges in Kilogramm mit Rotationszuschlag, dann die Koeffizienten zur Approximation von Reibungskräften nach der Beziehung

$$u_{reib}(v) = k_0 + k_1 v + k_2 v^2 \tag{8.1}$$

und schließlich das Sampling der geschwindigkeitsabhängigen maximalen Antriebskraft $u_{max}(v)$ an den Stellen $v = 0, 0.1, 0.2 \cdots$ Meter pro Sekunde in Newton.
Auch hier ein Beispiel:

```
#
# Beispiel eines Zugdaten-Files
#
# Masse mit Rotationszuschlag in Kilogramm
  876000
# Reibungskoeffizienten  k_0,k_1,k_2
  1000.0 0.0 10.3
# Approximation umax(v)
#      v[m/s]      umax[N]
       0.000000    400137.000000
       0.100000    400029.000000
       0.200000    399921.000000
       0.300000    399813.000000
       0.400000    399704.000000
       0.500000    399596.000000
       0.600000    399488.000000
       0.700000    399380.000000
       0.800000    399272.000000
       ...
```

8.1.5 Das Parameterfile

Zur Steuerung von numerischen und anderen Parametern kann beim Aufruf des Programms ein Parameterfile angegeben werden.
Dies ist ein ASCII-File mit Zeilen der Form:

```
parametername = wert
```

In dem Parameterfile müssen nicht alle Parameter angegeben werden, fehlende werden auf Default-Werte gesetzt.

Die Reihenfolge der Parameter ist beliebig, zu beachten ist jedoch, daß bei mehrfachen Auftauchen desselben Parameters der Wert des letzten Males des Auftauchens angenommen wird.
Möglich sind die im folgenden beschriebenen Parameter.

plot_para Ausgabe des Parameterfiles (Default=1)

plot_wegliste_vor_glaetten Ausgabe der Wegliste vor dem Glätten der Abwärtssprünge von $vmax(s)$ (Default=1).

plot_wegliste_nach_glaetten Ausgabe der Wegliste nach dem Glätten der Abwärtssprünge von $vmax(s)$, d.h. mit den zusätzlich eingefügten Absprungintervallen und den bestimmten Bremskurven (Default=1).

plot_vmax_vor_glaetten Erzeugen eines Gnuplot-Datenfiles mit der vorgegebenen zulässigen Maximalgeschwindigkeit (Default=1).

plot_vmax_nach_glaetten Erzeugen eines Gnuplot-Datenfiles mit der geglätteten zulässigen Maximalgeschwindigkeit (Default=1).

plot_bremskurve_aller Steuerparameter zur Generierung von Datenpunkten zum Plotten von Bremskurven. Es wird aller `plot_bremskurve_aller` Meter ein Datenpunkt erzeugt (Default=1.0 m).

plot_bremskurven Erzeugen eines Gnuplot-Datenfiles mit allen möglichen durch die Sicherungsblöcke bedingten Bremskurven (Default=1).

plot_vmax_rel Erzeugen eines Gnuplot-Datenfiles mit dem Minimum der 3 Restriktionen (Default=1).

v_inf Wert der zulässigen Höchstgeschwindigkeit, der für Dummyblöcke angenommen wird. Dies sollte ein Wert sein, der niemals zu einer Restriktion werden kann. (Default=500 km/h)

simu_stepsize Zeitschrittweite der Simulation (Default=1.0 s)

simu_maximal Maximale Simulationszeit, wird sie überschritten, so erfolgt ein Abbruch mit einer Exception. (Default=86400 s, entspricht 24 Stunden)

Wird kein Parameterfile angegeben, dann werden alle Parameter auf ihre Defaultwerte gesetzt.
Außerdem ist bei Angabe von Parametern darauf zu achten, daß diese auch sinnvoll sind, im Programm werden die angegebenen Parameter ohne weitere Kontrolle übernommen.

8.2 Erstellen des Programmes

Das Compilieren des Programmes erfolgt durch Aufruf von

```
make
```

im fzr2d Verzeichnis.
Eventuell sind im Makefile der Compiler und seine Optionen anzupassen.
Voreingestellt ist der Gnu-Compiler g++.
Desweiteren ist es möglich, folgende Maros zu definieren:

- `FZR2D_DEBUG` ... Allgemeine Debuginformationen
- `FZR2D_ZEITMESSUNG` ... Aktivierung der Zeitmessungen
- `FZR2D_SIMUSTEPS` ... Ausgabe von t, s, und v Daten während der Simulation

Sämtliche Ausgaben des Programmes erscheinen auf `stdout` bzw. im Unterverzeichnis `./aus`, eventuell vorhandene Files darin werden überschrieben.

8.3 Ablauf des Programms

Mit entsprechend vorbereiteten Eingabefiles, erfolgt der Aufruf des Programmes mit

```
./fzr2d globales_eingabefile [parameterfile]
```

Dies bedeutet, daß die Angabe des Parameterfiles optional ist, fehlt es, dann werden vordefinierte Default-Werte für alle Parameter verwendet. Zuerst werden die Eingabefiles eingelesen, und dann startet die Simulation. Ausgaben erfolgend entsprechend den beim Compilieren eingestellen Makros.
Zum Weiterverarbeiten der Ausgaben ist das im Anhang auf Seite 103 beschriebene Programm `fzr2d-outconv` hilfreich, mit ihm lassen sich aus den Ausgaben von fzr2d Gnuplot-Datenfiles erstellen.

Kapitel 9

Beispielrechnungen zu FZR2D

In diesem Kapitel sollen die Ergebnisse der Rechnung einiger Beispiele ausgewählter Streckenarten vorgestellt werden.

9.1 Hintereinanderfahren zweier Züge

Betrachten wir das Hintereinanderfahren zweier Züge auf ein und derselben Strecke.
Der vordere Zug wird dabei zuerst starten, und der zweite mit etwas Zeitversatz.
Es ist klar, daß der vorn fahrende Zug durch den hinteren nicht beeinflußt wird, umgekehrt jedoch der hintere immer in einem bestimmten Abstand zum vorderen bleiben muß.
Dies wird durch daß Sicherungskonzept der Blocksicherung erreicht.
Um das Beispiel etwas interessanter zu gestalten, sind bei dem hinteren Zug noch folgende Änderungen im vergleich zum Vorderen vorgenommen worden:

- Im Fahrzeug-Datenfile ist die Masse des Zuges herabgesetzt, so daß er schneller beschleunigen kann.
- Im Laufweg-Datenfile ist auf der gesamten Strecke (außer im Haltepunkt) die streckenbezogene zulässige Maximalgeschwindigkeit auf 200 km/h hochgesetzt.

Siehe dazu die beiden Bilder mit den bereits geglätteten Geschwindigkeitsrestriktionen in den Bildern 9.1 und 9.2
Der hintere Zug wird also durch diese Änderungen den vorderen Zug sehr schnell einholen und muß durch das Sicherungskonzept aufgehalten werden.
Am besten erkennt man das Wirken der Sicherung in dem $v(s)$ - Verlauf des hinteren Zuges (Bild 9.4), dort wechseln sich ständig Beschleunigungs- und Bremsphasen miteinander ab, ohne daß die streckenbezogene Maximalgeschwindigkeit erreicht wird.
Die Bremsphasen gehen dabei nicht immer bis zum Wert $v = 0$, was daran liegt, daß beim Annähern an das Ende des (noch gesperrten) Sicherungsblockes erst einmal verzögert werden muß, wenn jedoch der Block durch den vorderen Zug freigegeben wird, dann kann sofort wieder beschleunigt werden.
Der vordere Zug hingegen wird nur durch seine streckenbezogene Maximalgeschwindigkeit restringiert (Bild 9.4).

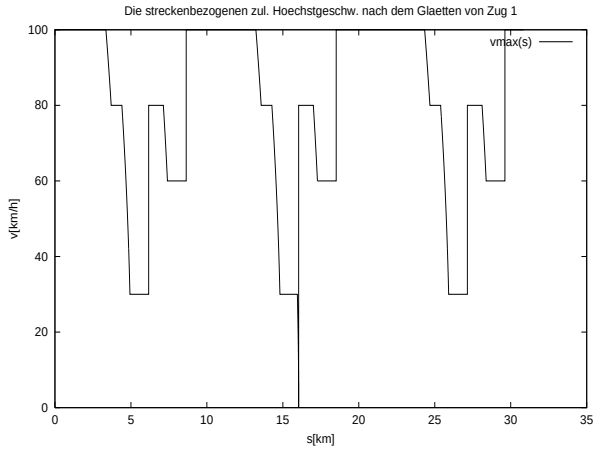

Abbildung 9.1: Zulässige Höchstgeschwindigkeit des vorderen Zuges

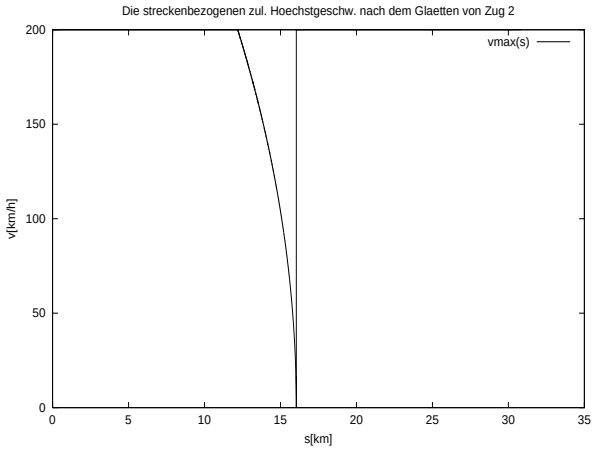

Abbildung 9.2: Zulässige Höchstgeschwindigkeit des hinteren Zuges

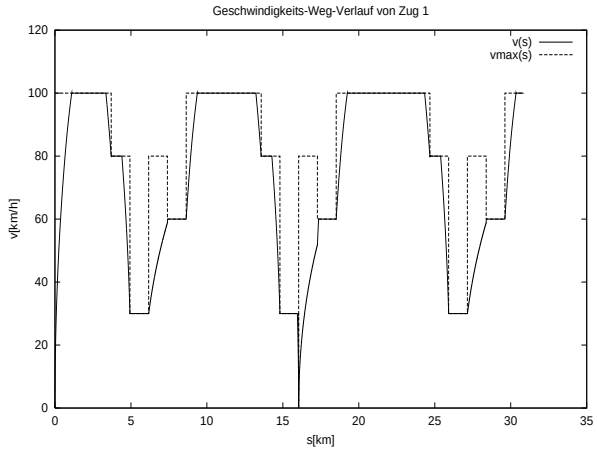

Abbildung 9.3: Geschwindigkeitsverlauf des vorderen Zuges

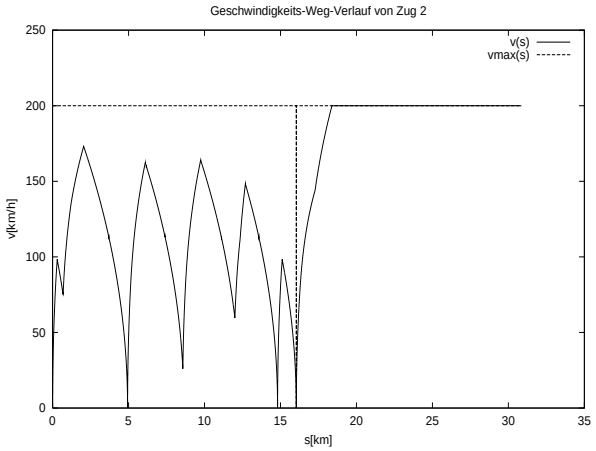

Abbildung 9.4: Geschwindigkeitsverlauf des hinteren Zuges

Außerdem wird das Funktionieren der Sicherung im $t(s)$ Diagramm (Bild 9.5) deutlich, denn wenn sich im $t(s)$-Verlauf die Trajektorien der beiden Züge in einem Punkt $[s_0, t_0]^\tau$ schneiden, so bedeutet dies, daß beide Züge zur gleichen Zeit am gleichen Ort waren. Dies kommt einem Unfall im Punkt $[s_0, t_0]^\tau$ gleich.
Man erkennt auch gut, wie der hintere Zug die Haltezeit des vorderen Zuges abwarten muß, dann in den Bahnhof einfährt und seine eigene vorgegebene Haltezeit dort wartet.

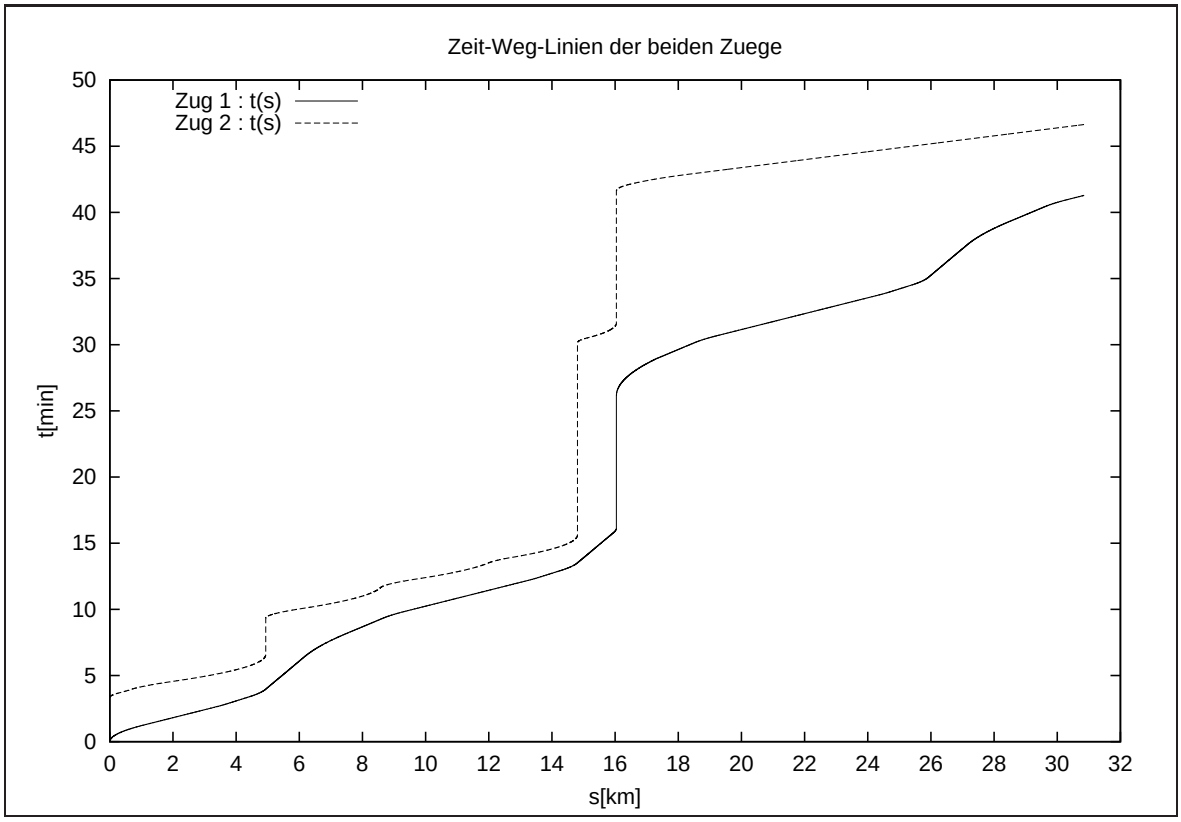

Abbildung 9.5: Zeit-Weg-Verlauf der beiden Züge

9.2 Fahrt über eine Weiche

Dies ist ein Beispiel für die Fahrt zweier Züge über eine Weiche.

Die Strecke ist in Bild 9.6 skizziert, die Nummern der relevanten Sicherungsblöcke sind dabei angegeben.

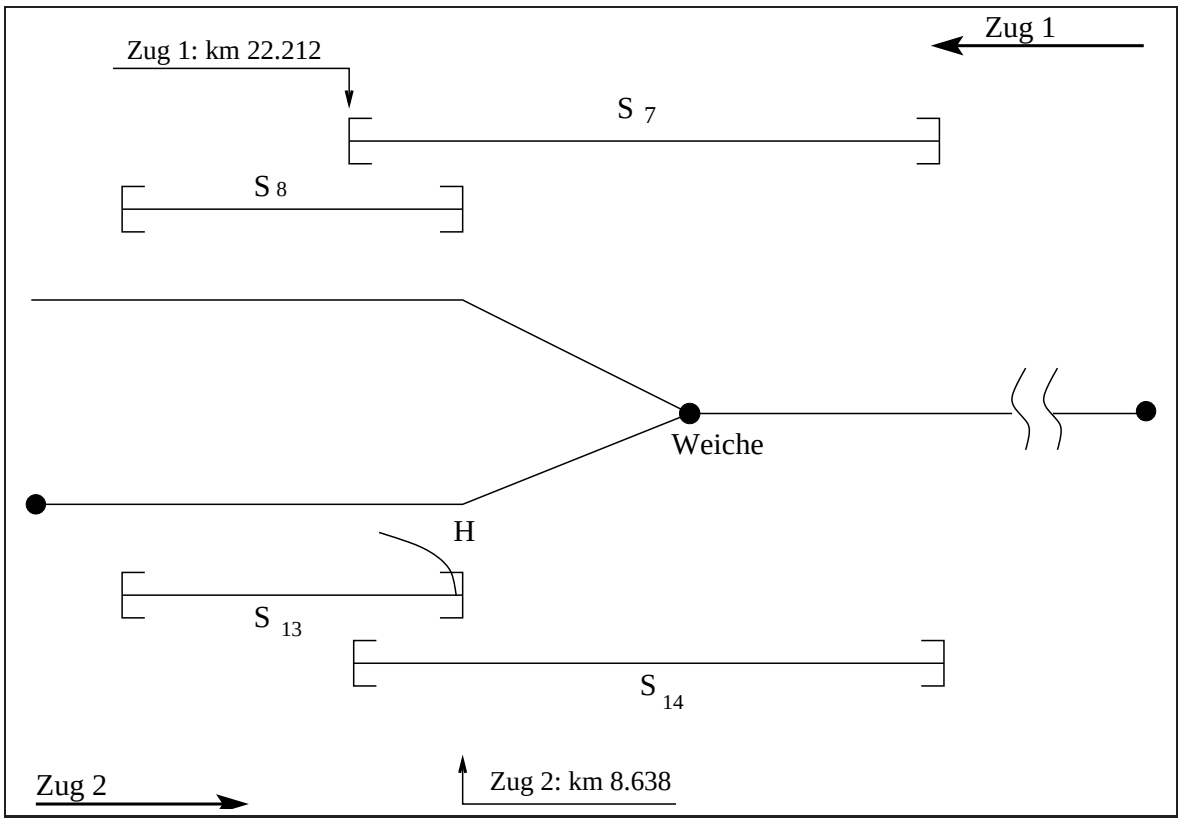

Abbildung 9.6: Skizze zur Fahrt über eine Weiche

Beide Züge starten gleichzeitig, Zug 1 fährt von rechts nach links oben, Zug 2 von links unten nach rechts.

Die streckenbezogene Höchstgeschwindigkeit liegt bei beiden Zügen im Abschnitt mit der Weiche bei 60 Kilometer pro Stunde, sonst bei 100.

Die Züge benutzen also den rechten Gleisabschnitt beide gemeinsam in entgegengesetzter Richtung.

Zug 1 soll dabei den Vorrang haben, d.h. die Weiche zuerst passieren können, anderenfalls würden die Züge auf dem eingleisigen Stück einander gegenseitig zum Halten zwingen.

Dies wird durch entsprechend definierte PMA und Initialbelegungen von β erreicht, es gilt

$$\texttt{sperr(j)} = \emptyset \qquad \forall j \tag{9.1}$$

$$\texttt{entsperr(j)} = \begin{cases} \{13\} & j = 7 \\ \emptyset & \text{sonst} \end{cases} \tag{9.2}$$

$$\beta_j = \begin{cases} 1 & j = 13 \\ 0 & \text{sonst} \end{cases} \tag{9.3}$$

wobei hier, der Einfachheit halber, nur der Teil mit der Weiche gesichert wurde.
In der Praxis müssen natürlich alle Sicherungsblöcke entsprechend gesichert werden, dies hat aber nur Einfluß für unplanmäßige Fahrten, um Unfälle mit anderen Zügen zu verhindern.
Da wir in diesem Beispiel nur die beiden Züge haben, reicht dies so aus.
Wie auch in der Skizze ersichtlich, liegt das Ende von S_{13} bei Kilometer 8.636, relativ zum Laufweg des zweiten Zuges und es liegt das Ende von S_7 bei Kilometer 22.212, relativ zum Laufweg des ersten Zuges.
Dies bedeutet, der zweite Zug darf auf seiner Strecke den Kilometer 8.636 erst dann überfahren, wenn der erste Zug seinen Kilometer 22.212 passiert hat.
Wie man im $t(s)$-Verlauf der beiden Züge (Bild 9.7) erkennen kann, funktioniert dies auch, denn Zug 2 wartet am Kilometer 8.636 bis Zug 1 seinen Kilometer 22.212 passiert hat.
Daß die $t(s)$-Kurven am Anfang aufeinanderliegen bedeutet nicht, daß die Züge kollidieren, denn die Wegstrecke ist relativ zum Zug angegeben. Das erste Stück des Weges von Zug 1 ist auf einem anderen Gleis als das von Zug 2, somit ist Kollision ausgeschlossen.
Um Kollisionen erkennen zu können, müssen bei deartigen Verläufen die Streckenkilometer über den Kreuzungspunkt zueinander relativiert werden.
Bilder 9.9 und 9.10 zeigen den Geschwindigkeits-Weg-verlauf der beiden Züge, man erkennt den Halt von Zug 2 an seinem Streckenkilometer 8.636.

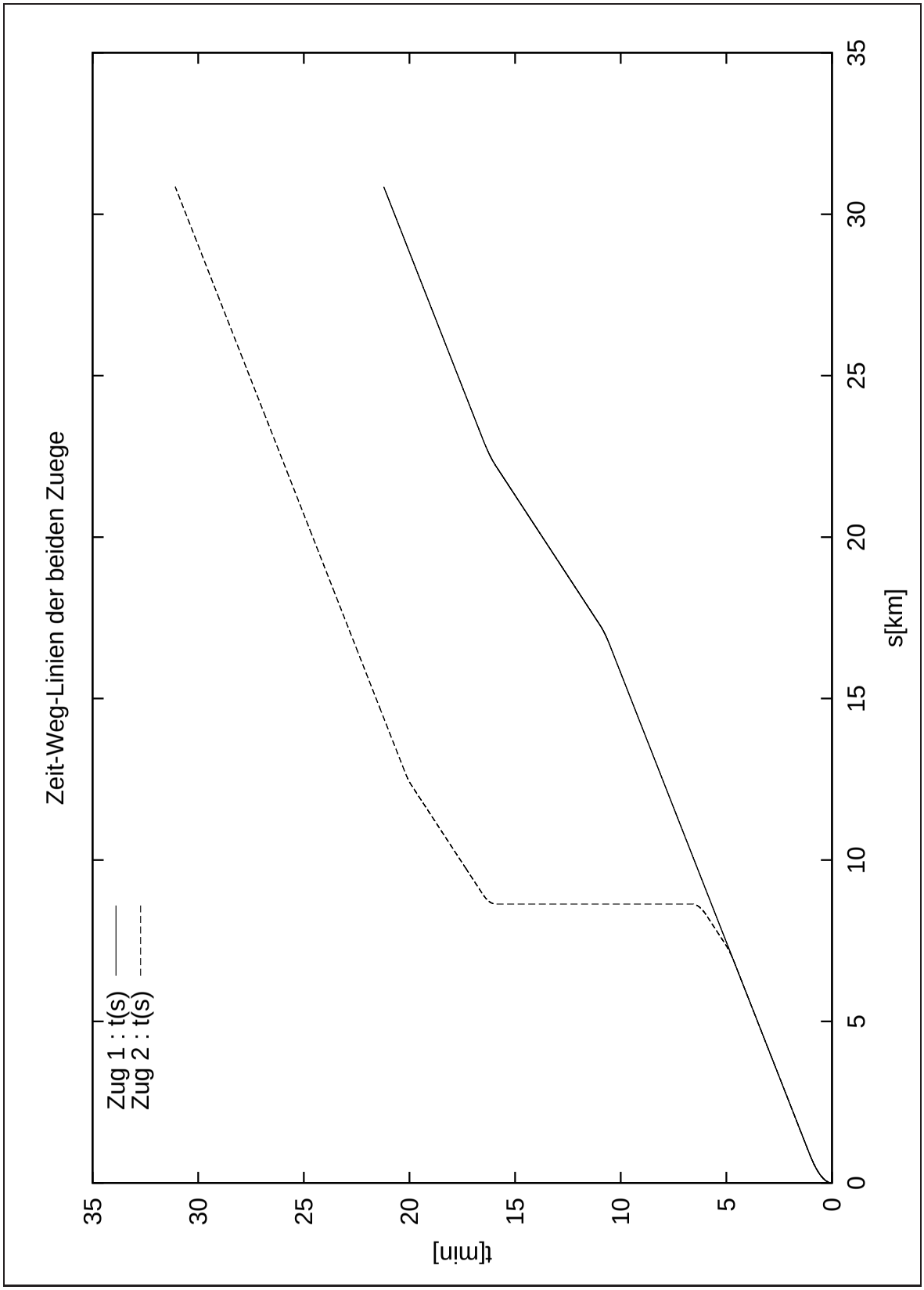

Abbildung 9.7: Zeit-Weg-Verlauf der beiden Züge in zugbezogener Kilometrierung

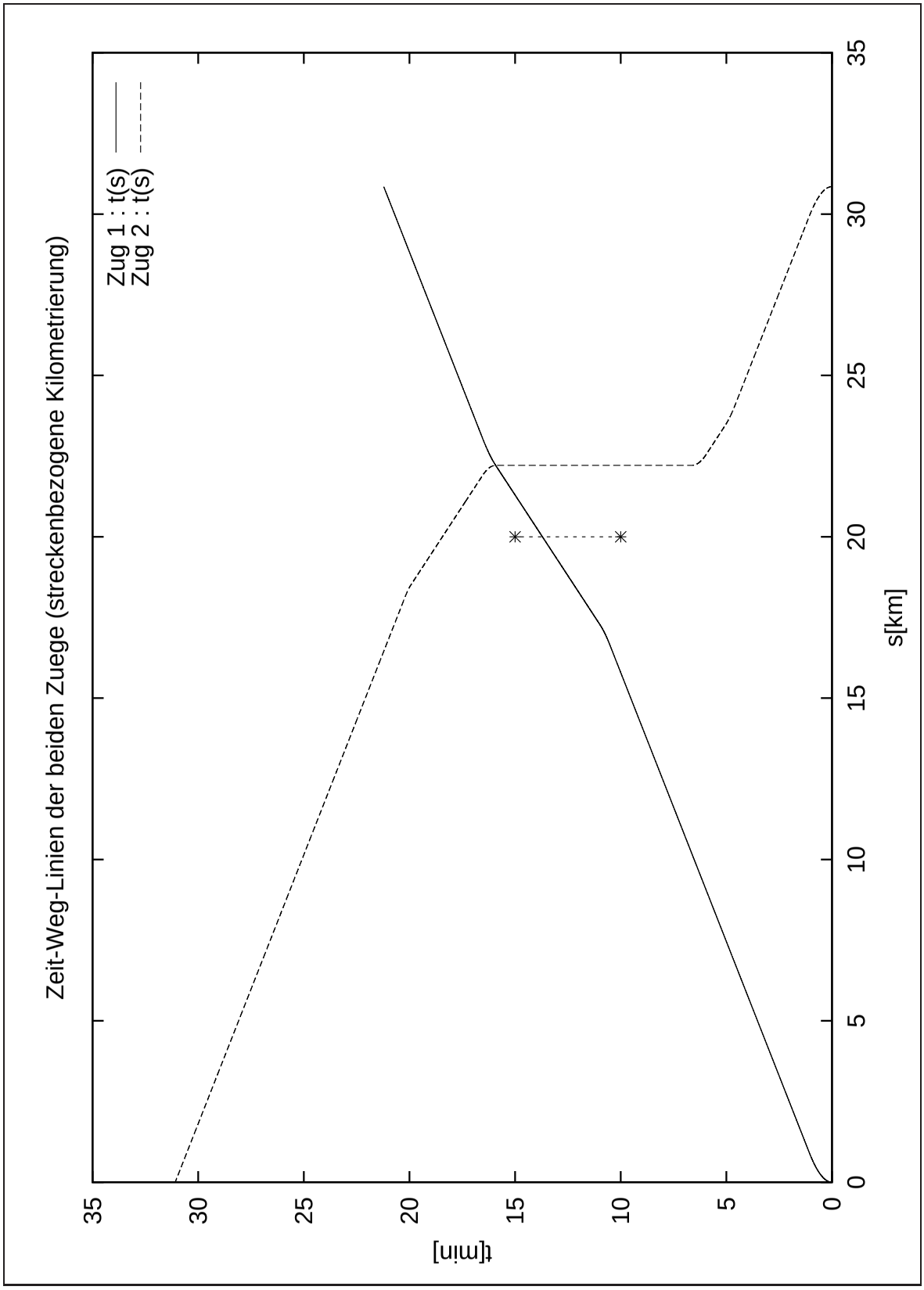

Abbildung 9.8: Zeit-Weg-Verlauf der beiden Züge in streckenbezogener Kilometrierung

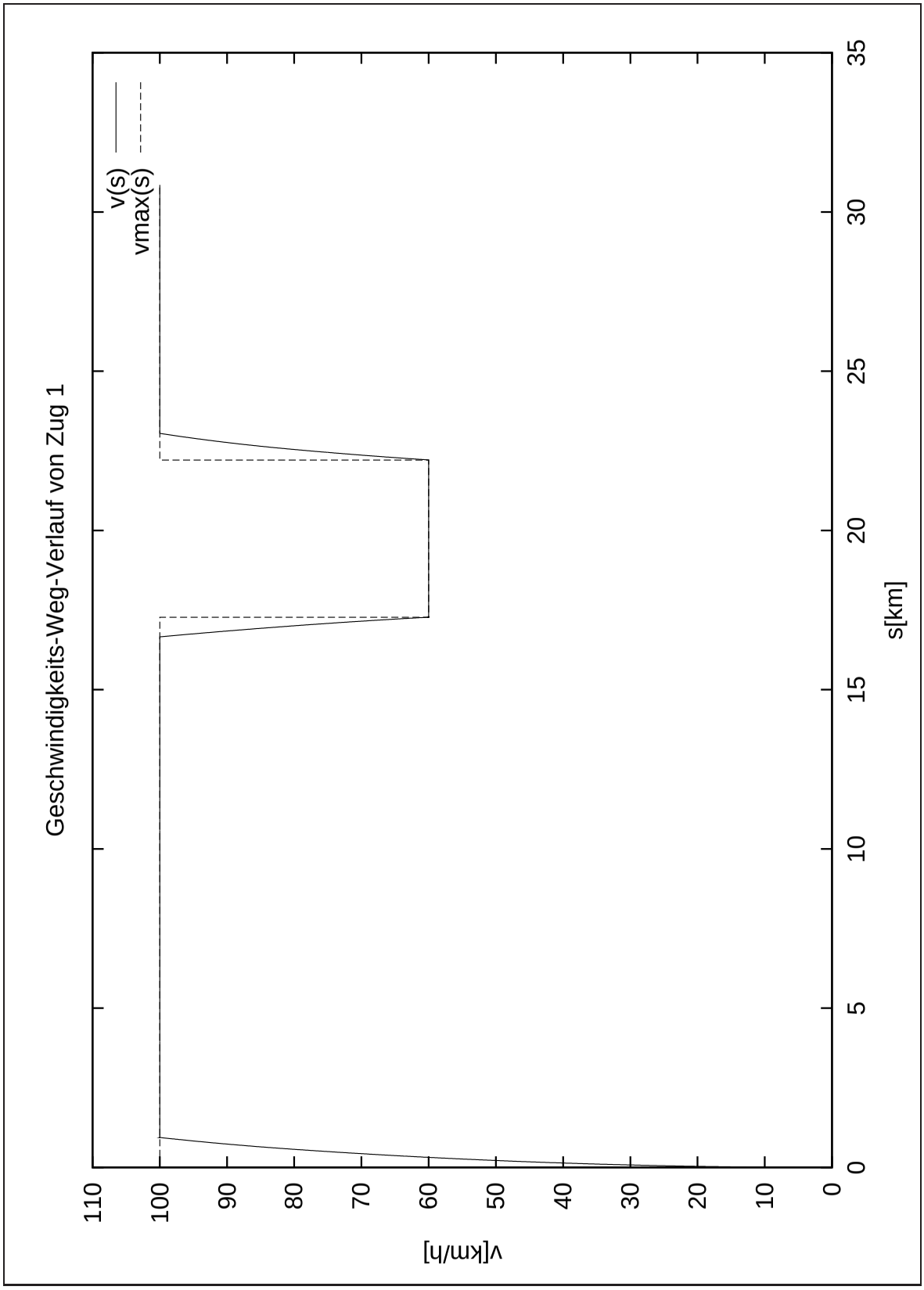

Abbildung 9.9: Geschwindigkeits-Weg-Verlauf des ersten Zuges

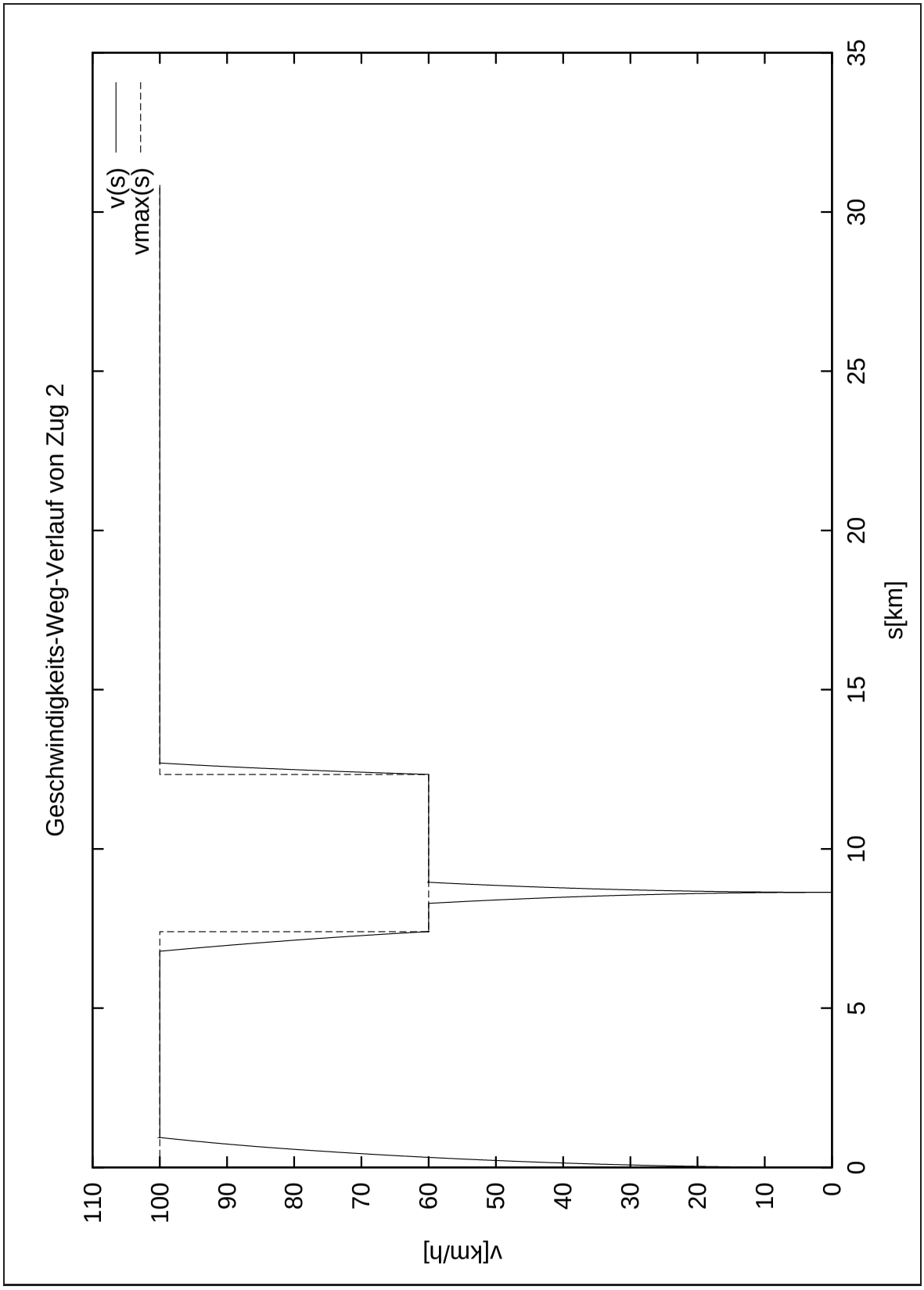

Abbildung 9.10: Geschwindigkeits-Weg-Verlauf des zweiten Zuges

9.3 Laufzeitvergleiche

In diesem Abschnitt sollen ein paar Vergleiche zur Rechnung und Laufzeit mit unterschiedlichen Simulationsschrittweiten angeführt werden.
Bei dem gerechneten Beispiel fährt wieder ein schneller Zug hinter einem langsamen Zug auf der selben Strecke.
Der hintere Zug startet dabei 10 Minuten nach dem vorderen. Die Strecke eines jeden Zuges ist 450 Kilometer lang und setzt sich aus 299 Wegabschnitten zusammen.
Es existieren 200 Sicherungsblöcke, die zu je 100 auf die beiden Laufwege aufgeteilt sind.
Das Beispiel wurde auf einer Intel-Pentium-III Maschine mit Taktfrequenz von 800 Megahertz unter SuSE-Linux 7.0 gerechnet.
Die Zeit zum Einlesen aller Datenfiles umfaßt etwa 0.25 Sekunden, das Beispiel wurde mit verschiedenen Simulationsschrittweiten von 0.001 bis 20 Sekunden gerechnet.
Die Ergebnisse sind in der folgenden Tabelle dargestellt.
In der ersten Spalte ist die Schrittweite der Simulation in Sekunden aufgeführt, dann die für die Züge ermittelten Ankuftszeiten am Ende der Strecke.
Für jeden Zug wird die Rechenzeit für die Simulation der Fahrt einzeln erfaßt, aufgeführt in den letzten beiden Spalten.
Die Spalte Rechnen umfaßt die Zeit für die Bestimmung der Bremskurven sowie der Simulation aller Züge, also die eigentliche Rechenzeit des gesamten Programmes.

Schrittweite	Errechnete Ankunftszeiten		Rechenzeit	Simulationszeiten	
	ZUG 1	ZUG 2		ZUG 1	ZUG 2
[*Sekunden*]	[*Sekunden*]	[*Sekunden*]	[*Sekunden*]	[*Sekunden*]	[*Sekunden*]
0.001	28282.80	28794.20	184.10200	65.38240	74.03380
0.002	28282.80	28794.20	91.18990	32.52510	36.39770
0.003	28282.80	28794.20	60.79960	21.67300	24.26890
0.004	28282.80	28794.20	45.60550	16.26770	18.20170
0.005	28282.80	28794.20	36.48820	13.00240	14.57720
0.006	28282.80	28794.20	30.40500	10.84860	12.13240
0.007	28282.80	28794.20	26.10670	9.29813	10.43260
0.008	28282.80	28794.20	22.81470	8.12504	9.09888
0.009	28282.80	28794.20	20.27700	7.22326	8.10441
0.010	28282.70	28794.10	18.25100	6.50096	7.29321
0.020	28282.70	28794.10	9.13476	3.24951	3.65637
0.030	28282.60	28794.00	6.07927	2.16806	2.42563
0.040	28282.50	28793.90	4.57622	1.62540	1.82087
0.050	28282.50	28793.90	3.64830	1.30030	1.45504
0.060	28282.40	28793.80	3.05671	1.08382	1.21349
0.070	28282.40	28793.80	2.60736	0.92919	1.03980
0.080	28282.20	28793.70	2.28076	0.81228	0.90958
0.090	28282.20	28793.60	2.02838	0.72341	0.80832
0.100	28282.20	28793.60	1.84118	0.65075	0.74299
0.200	28281.40	28792.70	0.91436	0.32534	0.36460
0.300	28280.30	28791.70	0.61092	0.21733	0.24311

Schrittweite	Errechnete Ankunftszeiten		Rechenzeit	Simulationszeiten	
	ZUG 1	ZUG 2		ZUG 1	ZUG 2
[*Sekunden*]	[*Sekunden*]	[*Sekunden*]	[*Sekunden*]	[*Sekunden*]	[*Sekunden*]
0.400	28280.50	28791.60	0.45907	0.16325	0.18249
0.500	28279.40	28790.30	0.36763	0.13056	0.14621
0.600	28279.30	28790.20	0.30713	0.10912	0.12180
0.700	28277.00	28787.80	0.26333	0.09332	0.10439
0.800	28277.50	28788.30	0.23081	0.08168	0.09169
0.900	28276.40	28787.00	0.20603	0.07301	0.08139
1.000	28274.90	28785.50	0.18555	0.06551	0.07348
1.500	28272.50	28782.80	0.12468	0.04392	0.04918
2.000	28264.70	28774.40	0.09427	0.03315	0.03688
2.500	28264.20	28773.90	0.07616	0.02671	0.02966
3.000	28261.70	28771.40	0.06405	0.02219	0.02501
3.500	28261.70	28770.40	0.05516	0.01909	0.02131
4.000	28251.60	28758.50	0.04881	0.01678	0.01884
4.500	28252.10	28758.20	0.04362	0.01505	0.01678
5.000	28259.90	28763.40	0.03975	0.01375	0.01507
5.500	28251.90	28754.30	0.03600	0.01227	0.01372
6.000	28252.30	28758.60	0.03377	0.01154	0.01272
6.500	28228.60	28731.70	0.03127	0.01059	0.01180
7.000	28232.40	28733.90	0.02911	0.00983	0.01088
7.500	28230.80	28729.40	0.02749	0.00921	0.01026
8.000	28237.80	28733.80	0.02603	0.00881	0.00965
8.500	28219.70	28717.30	0.02454	0.00821	0.00913
9.000	28214.80	28706.20	0.02342	0.00777	0.00872
9.500	28226.20	28719.70	0.02248	0.00741	0.00838
10.000	28214.50	28700.20	0.02146	0.00704	0.00793
11.000	28291.50	28785.30	0.01981	0.00650	0.00724
12.000	28261.30	28752.90	0.01843	0.00597	0.00667
13.000	28276.10	28756.90	0.01737	0.00559	0.00621
14.000	28284.40	28775.20	0.01637	0.00536	0.00585
15.000	28274.10	28734.40	0.01542	0.00488	0.00553
16.000	28231.30	28714.90	0.01475	0.00463	0.00528
17.000	28282.70	28716.20	0.01394	0.00439	0.00494
18.000	28251.50	28708.00	0.01342	0.00419	0.00468
19.000	28294.20	28739.90	0.01328	0.00402	0.00486
20.000	28371.30	28819.80	0.01249	0.00385	0.00436

Wie man an den Daten erkennt, ist bei sinnvoller Wahl der Simulationsschrittweite ein guter Kompromiß zwischen Genauigkeit und Laufzeit der Rechnung möglich.
Für praktische Zwecke ist es nämlich nicht nötig bei einer Fahrzeit des Zuges von etwa 8 Stunden seine Ankunftszeit auf die Sekunde genau vorauszuberechnen.
Zum Rechnen der Testreihen ist im Anhang auf Seite 104 das Hilfsprogramm beschrieben.

9.4 Eingleisige Strecke mit Ausweichstelle

Hier betrachten wir eine eingleisige Strecke mit Ausweichstelle, die im Gegenverkehr befahren werden soll.
So eine Strecke existiert zum Beispiel zwischen meiner Heimatstadt Stollberg und Chemnitz.
Wie im Bild 9.11 dargestellt, fährt Zug 1 von links über den oberen Teil des Ausweichstückes nach rechts und Zug 2 über den unteren Teil von rechts nach links.

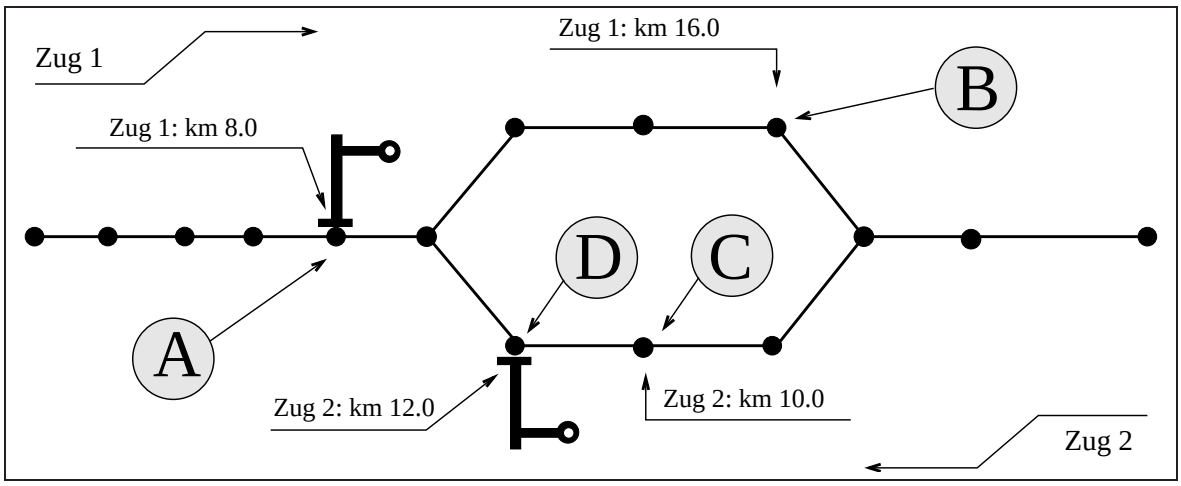

Abbildung 9.11: Skizze zur Fahrt im Gegenverkehr

Zunächst haben beide Züge ein Haltesignal in ihrem Laufweg, für Zug 1 bei seinem Kilometer 8 (Punkt A) und für Zug 2 bei seinem Kilometer 12 (Punkt D).
Diese beiden Signale sind so geschaltet, daß das Signal im Punkt A auf grün geht, wenn Zug 2 den Punkt C passiert und analog wird das Signal im Punkt D umgeschaltet, wenn Zug 1 den Punkt B passiert.
Mit diesem Konzept hat Zug 2 die Einfahrtsberechtigung in die Ausweichstelle, kann diese aber erst verlassen, wenn Zug 1 den Punkt B passiert hat.
Somit können die eingleisigen Stücke der Strecke immer nur von einem Zug benutzt werden.
Betrachtet man den Weg-Zeit-Verlauf der beiden Züge (Bild 9.12), so sieht man, daß auch dieses Konzept vom Programm korrekt modelliert wird, Zug 1 fährt bis zum Punkt A , wartet bis Zug 2 den Punkt C passiert hat, fährt dann wieder los, währenddessen Zug 2 in D hält, bis Zug 1 den Punkt B passiert hat.
Um die beiden Schaltpunkte bei B und C besser zu erkennen, sind diese im Bild durch gestrichelte Linien mit ihrer korrespondierenden Fahrzeit verbunden.

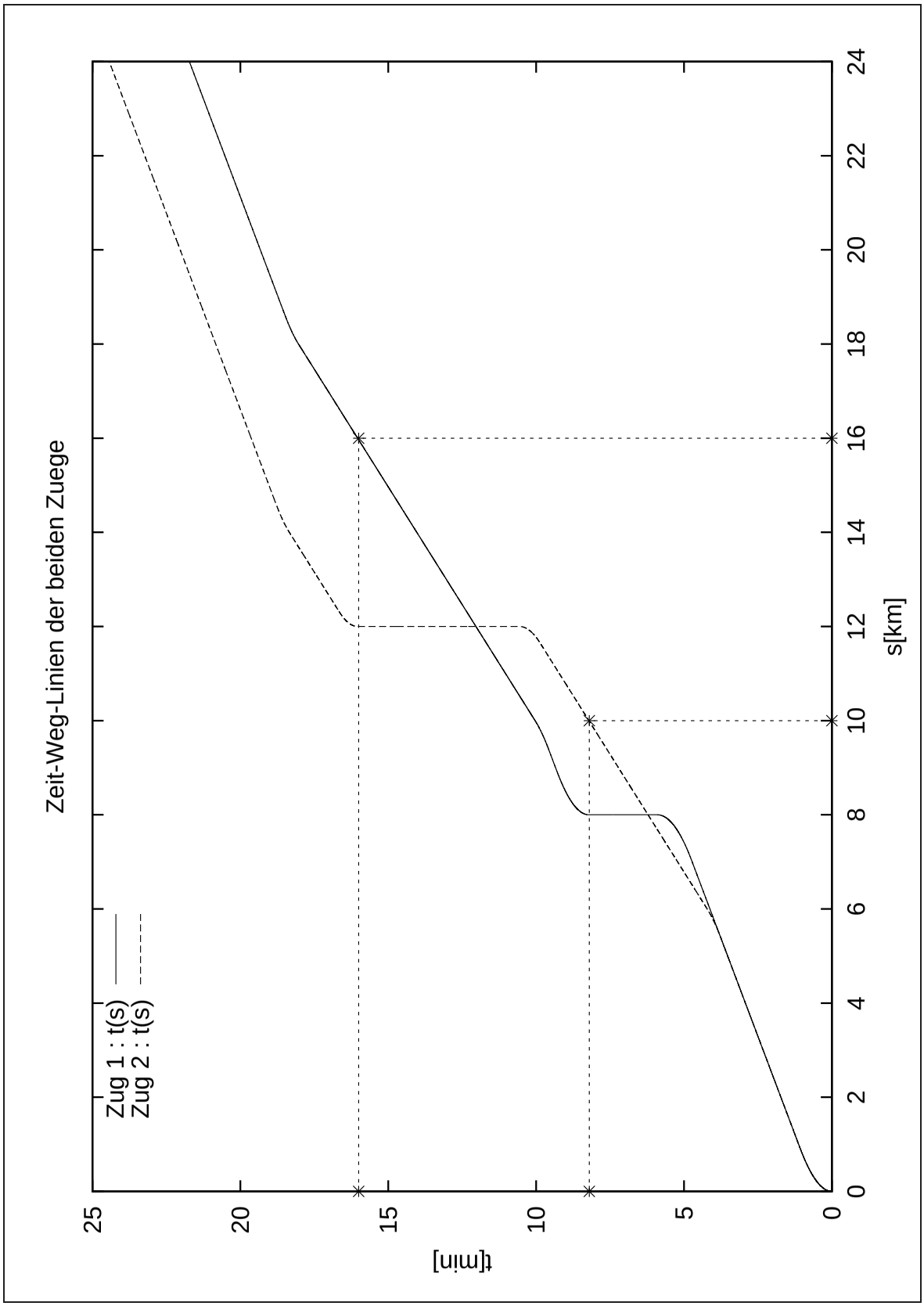

Abbildung 9.12: Zeit-Weg-Verlauf der beiden Züge im Beispiel der eingleisigen Strecke in zugbezogener Kilometrierung

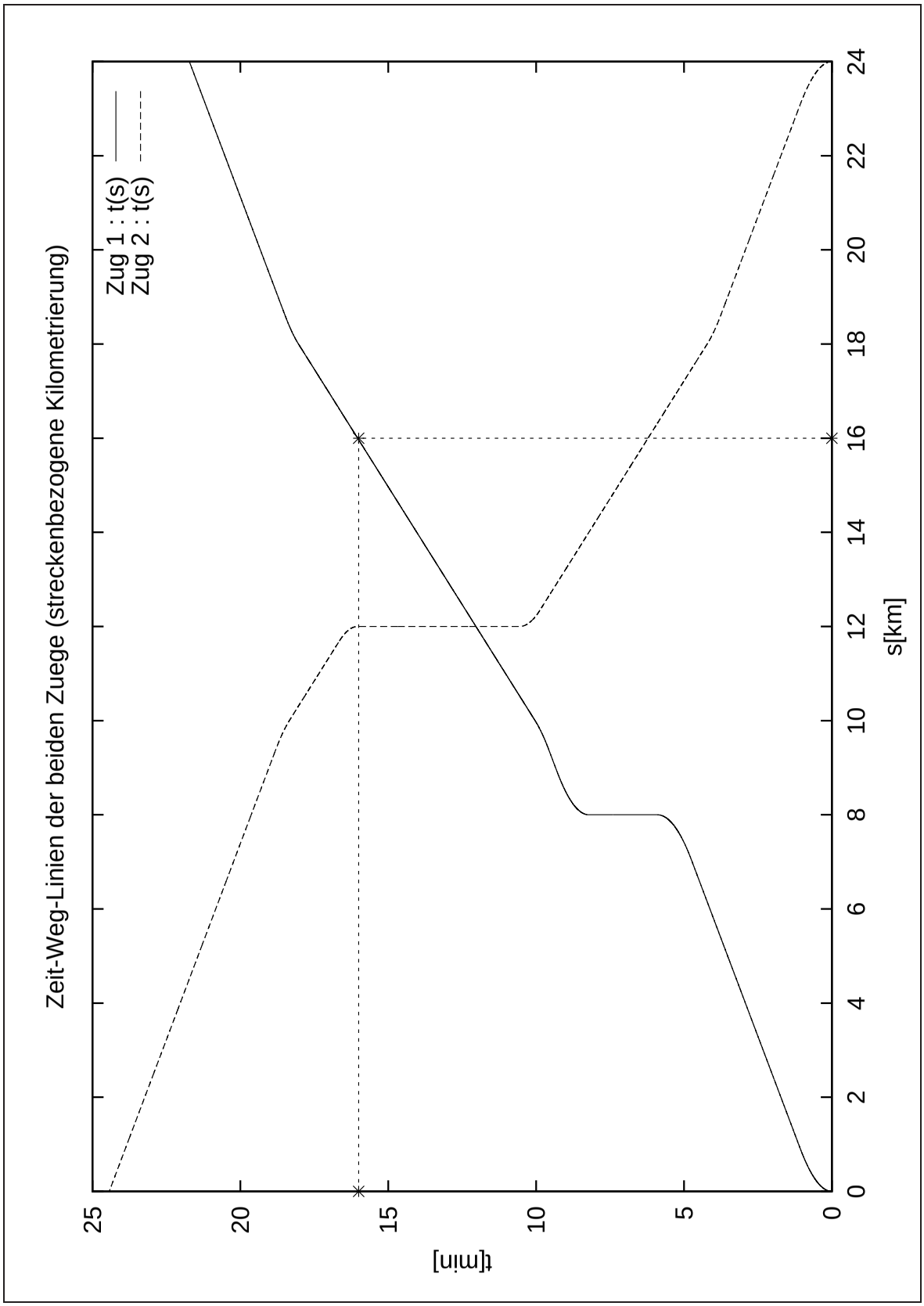

Abbildung 9.13: Zeit-Weg-Verlauf der beiden Züge im Beispiel der eingleisigen Strecke in streckenbezogener Kilometrierung

Kapitel 10

Vergleich verschiedener Wartezeiten

In den vorangegangenen Beispielen hat sich gezeigt, daß bei der Hintereinanderfahrt zweier Züge der hintere Zug vom vorderen dominiert wird.
Dabei stellen sich die beiden folgenden Fragen:

1. Welchen Einfluß hat die Startzeit des hinteren Zuges auf seine Ankunftszeit ?
2. Wie beeinflussen unterschiedliche Startzeiten den Energieverbrauch des hinteren Zuges ?

Eine Betrachtung der Weg-Zeit-Linien der beiden Züge legt die Vermutung nahe, daß die Ankunftszeiten des hinteren Zuges von seiner Startzeit unabhängig sind, solange er es schafft, den vorderen Zug einzuholen.
Die Fahrweise des hinteren Zuges nach dem Einholen ist, entsprechend dem bang-bang-Prinzip, ein ständiger Wechsel zwischen Beschleunigen und Bremsen.
Man vermutet, daß dies energetisch sehr unökonomisch ist.
Zum Überprüfen dieser Vermutungen eignet sich eine Testreihe, in der die Startzeit des hinteren Zuges systematisch erhöht wird und dabei jeweils seine Ankunfts- und Fahrzeit gemessen wird.
Desweiteren sei folgende, aus den diskreten Werten für Zeit und Geschwindigkeit abgeleitete Energiesumme definiert

Definition 12 (Energiesumme) *Seien die Werte t_i und v_i $(i = 1 \cdots n)$ gesampelte Meßpunkte des Geschindigkeits-Zeit-Verlaufes einer simulierten oder realen Zugfahrt. Dann soll die Größe*

$$E := \sum_{i=1}^{n-1} \frac{(v_{i+1} - v_i)^2}{t_{i+1} - t_i} \tag{10.1}$$

die Energiesumme bezeichnen.

Bemerkung 5 *Die Energiesumme stellt eine einfache numerische Approximation des über die Beschleunigung definierten Energiefunktionals*

$$J(a(t)) := \int_0^t a(t)^2 dt \tag{10.2}$$

dar. Dabei werden Brems- und Beschleunigungsphasen gleichermaßen berücksichtigt.

Es zeigt sich nämlich, daß die Ankunftszeit des hinteren Zuges bis zu einer gewissen Wartezeit konstant bleibt und dann (wenn der vordere Zug nicht mehr eingeholt werden kann) monoton wachsend ist.
Diese Dauer der Wartezeit sei im weiteren mit t_w^* bezeichnet.
Analog ergibt sich, daß die reine Fahrzeit mit wachsender Wartezeit bis zu einer unteren Schranke fällt und ab der Wartezeit t_w^* dann auf diesem Wert bleibt. (Dies ist die Fahrzeit, die vom hinteren Zug mindestends benötigt wird, sie ergibt sich auch, wenn er die Strecke allein befährt.)
Für den Verlauf der Ankunfts- und Fahrzeiten siehe Bilder 10.1 bis 10.4, dort ist jeweils ein Verlauf über einen längeren Zeitraum und ein Zoom um den Punkt t_w^* herum dargestellt.

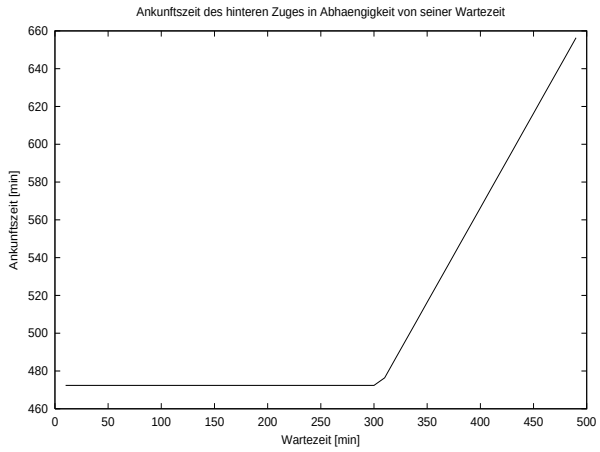

Abbildung 10.1: Ankunftszeit

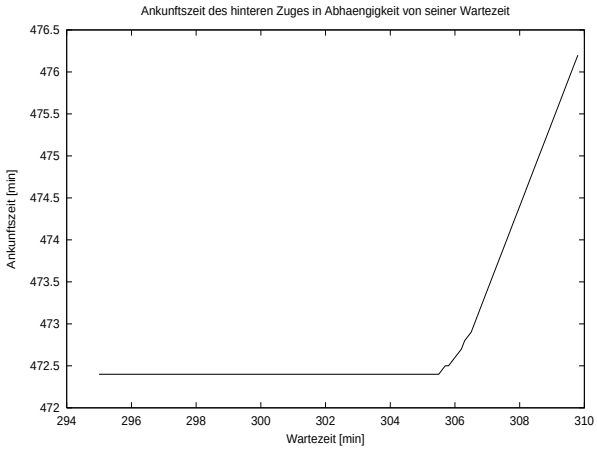

Abbildung 10.2: Zoom um den Punkt t_w^*

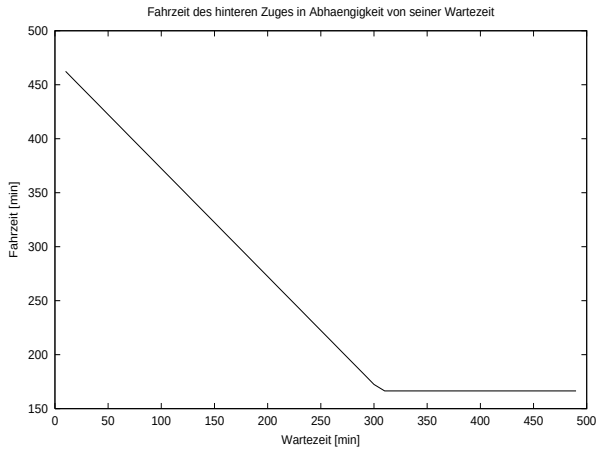

Abbildung 10.3: Fahrzeit

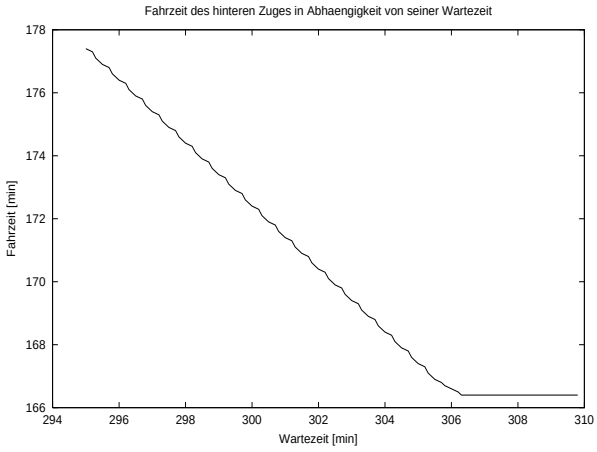

Abbildung 10.4: Zoom um den Punkt t_w^*

Es ergibt sich also, daß die Ankunftszeit des hinteren Zuges, bis zur Wartezeit t_w^* nicht von seiner Startzeit nicht abhängt.
Betrachtet man jedoch die Energiesumme, so zeigt sich, daß diese mit wachsender Wartezeit fällt, umd dann ab dem Punkt t_w^* auf einen konstanten Minimalwert zu gehen. (Bilder 10.5 und 10.5)

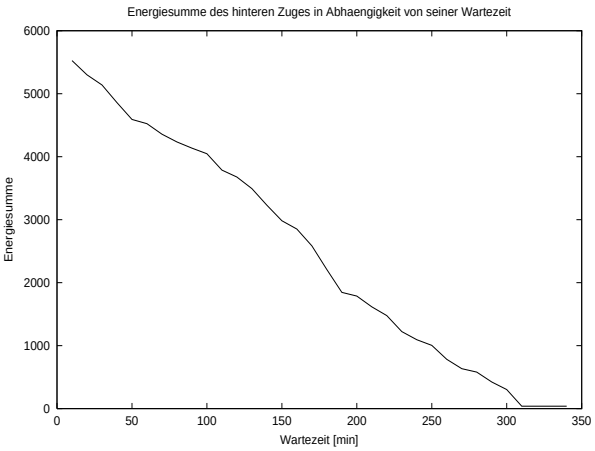

Abbildung 10.5: Energiesumme

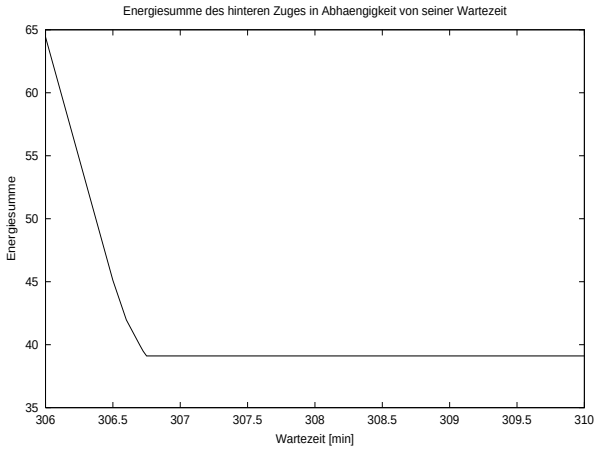

Abbildung 10.6: Zoom um den Punkt t_w^*

Man erkennt, daß es ökonomischer ist, nicht sofort hinterherzufahren, sondern den hinteren Zug erst eine gewisse Zeit warten zu lassen.
Als optimale Wartezeit ergibt sich dabei der ermittelte Umschlagpunkt t_w^*.
Zur Durchführung und Auswertung der Testrechnungen existieren einige Scripte und Hilfsprogramme, die im Anhang auf Seite 104 beschrieben sind.

10.1 Weiche mit und ohne Zugfolgeregelung

Dieser Abschnitt stellt die Beispiele zu der im Abschnitt 6.4.2 besprochenen Problematik der Einmündung zweier Züge in ein eingleisiges Streckenstück dar.
Die Weiche befindet sich dabei am Kilometer 13.5, d.h. vor diesem Punkt befinden sich die beiden Züge auf getrennten Gleisen, danach auf dem selben Gleis.
Es werden die folgenden 4 Testrechnungen mit unterschiedlichen Startzeiten der beiden Züge einmal mit und einmal ohne Zugfolgeregelung an der Einmündung betrachtet.

1. Es besteht keine Zugfolgeregelung. Zug 1 startet zur Zeit 0 und Zug 2 nach 90 Sekunden.

2. Es besteht keine Zugfolgeregelung. Zug 1 startet nach 90 Sekunden und Zug 2 zur Zeit 0.

3. Es besteht eine Zugfolgeregelung, die Zug 1 die Weiche als ersten passieren läßt. Zug 1 startet zur Zeit 0 und Zug 2 nach 90 Sekunden.

4. Es besteht eine Zugfolgeregelung, die Zug 1 die Weiche als ersten passieren läßt. Zug 1 startet nach 90 Sekunden und Zug 2 zur Zeit 0.

Es ergeben sich dann die Ankunftsfolgen der beiden Züge wie zu erwarten als :

Test 1 : Zug 1 kommt zuerst an
Test 2 : Zug 2 kommt zuerst an
Test 3 : Zug 1 kommt zuerst an
Test 4 : Zug 1 kommt zuerst an

Die Weg-Zeit-Linien der beiden Züge sind in den folgenden Bildern dargestellt.

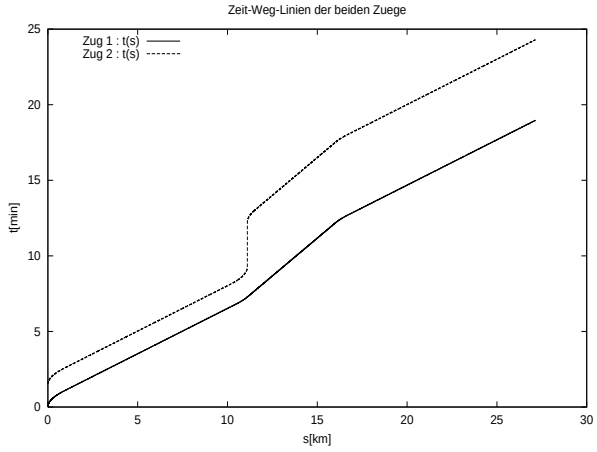

Abbildung 10.7: Zeit-Weg-Verlauf der beiden Züge zum Test 1

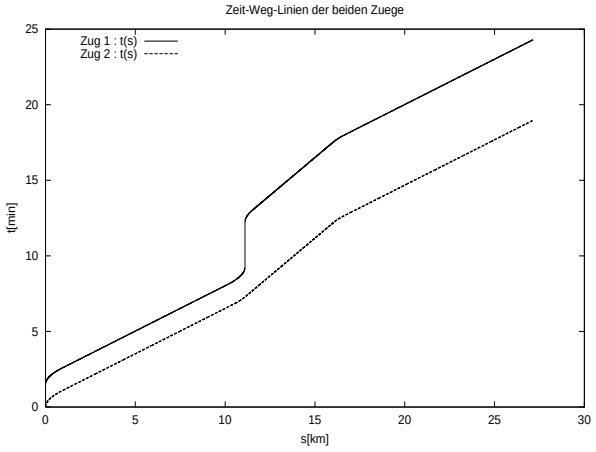

Abbildung 10.8: Zeit-Weg-Verlauf der beiden Züge zum Test 2

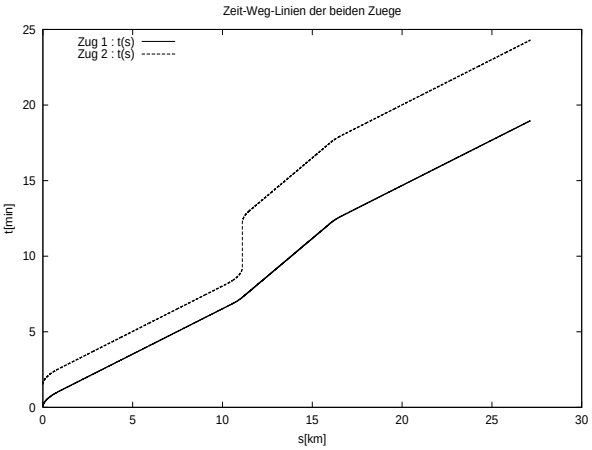

Abbildung 10.9: Zeit-Weg-Verlauf der beiden Züge zum Test 3

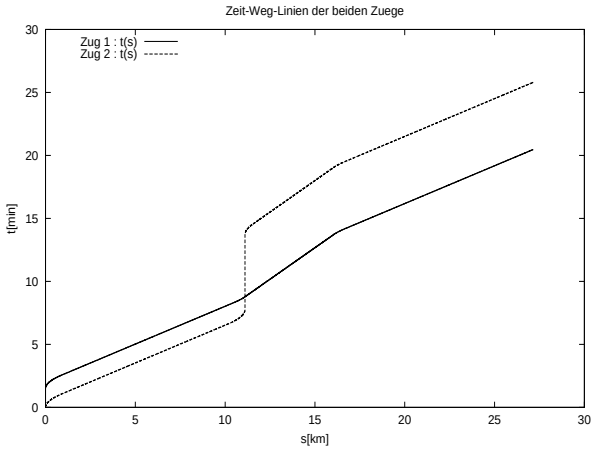

Abbildung 10.10: Zeit-Weg-Verlauf der beiden Züge zum Test 4

Kapitel 11

Ausblick

In den vorangegangenen Kapiteln wurde das Problem der zeitoptimalen Steuerung an einem einzelnen Zug sowie einem Netz aus mehreren Zügen betrachtet.
Es gibt weitere interessante Aspekte zur Bewertung der Optimalität der Steuerung von Zugfahrten, die sind unter anderem Energieoptimalität oder Komfortoptimalität für die Fahrgäste.
Es sind auch noch weitere Randbedingungen denkbar, z.B. Anschlußbedingungen verschiedener Züge, Fahrplanrestriktionen mit unterschiedlichen Wichtungen der Kosten beim Verletzen einer dieser Restriktionen.
Es wurde in den dargestellten Programmen, vor allem im Falle des einzelnen Zuges, auf kurze Rechenzeiten der Simulation Wert gelegt.
Die dabei erstellten Programme sind für sequentielle Abarbeitung auf einem Rechner konzipiert worden, die Algorithmen sind aber in gewissen Teilen parallelisierbar.
Bei der Simulation und Optimierung großer Netze werden nämlich Fragen der Speichertechnik und des Verwendens massiv paralleler Rechentechnik ein zu beachtender Aspekt.

Anhang A

Hilfs- und Konvertierprogramme

Es sind bei der Arbeit mit den Daten einige Hilfsprogramme entstanden, die hier kurz erläutert werden sollen.
Es handelt sich um C-Programme, die sich mit jedem C-Compiler, z. B. dem gcc durch Aufruf von

```
gcc program.c -o program
```

compilieren lassen.
In den Quelltexten finden sich auch noch weiterreichende Erläuterungen zu Parametern und anderem.

A.1 Programm wegform.c

Zum Konvertieren von Standard-Wegfiles im TLC-Format gibt es das Programm `wegform.c`
Diese Programm wird nach der Compilation gerufen als

```
wegform tlc_wegfile fzr_wegfile
```

und es konvertiert dann die Files entsprechend.
Zur Konvertierung ganzer Verzeichnisse existiert außerdem das Shell-Script `weg-all.sh`.

A.2 Programm graph2eps.c

Dieses Programm dient zur Darstellung zweidimensionaler Graphen als Encapsulated Postscript zur Weiterverarbeitung in anderen Dokumenten.
Eingabeparameter ist ein Graphenfile mit Knotenkoordinaten und einer Kantentabelle der Form

```
#
#  ... comments ...
#
```

```
# Anzahl Knoten
  4
# Knotenkoordinaten
# lfd.-Nummer  x  y
  1   0 0
  2  10 2
  ...
# Anzahl Kanten
  10
# Kantentabelle
# lfd.-Nummer; Anfangsknoten; Endknoten
  1 1 2
  2 2 3
   ...
# ENDE
```

Nach dem Aufruf des Programmes als

```
graph2eps graphenfile
```

erscheint die Ausgabe im File `graphenfile.eps`.
Die Breite des Randes und der Radius der Knotenmarkierungen wird durch Konstanten im Quelltext von graph2eps definiert und kann den eigenen Wünschen angepaßt werden.

A.3 Programm umaxhyperbel.c

Dieses Programm erzeugt ein Fahrzeugdaten-Eingabefile für die 1D-Fahrzeitrechnung mit einer geschwindigkeitsabhängigen maximalen Antriebskraft der Gestalt

$$u_{max}(v) = \frac{a_1}{a_2 v + a_3}$$

Dabei sind a_1, a_2 und a_3 frei wählbare Parameter, sie sind im Quelltext, wie auch die Masse und die Koeffizienten zur Approximation der Reibungskräfte zu setzen.

A.4 Programm umaxform.c

Umaxform dient zur Umwandlung von Datenfiles, in denen Samples für $u_{max}(v)$ in den Einkeiten Kilometer pro Stunde vs. Newton angegeben sind in Files mit den Einheiten Meter pro Sekunde vs. Newton.
Kommentarzeilen mit einem # in Spalte 1 werden dabei kopiert.
Die Handhabung des Programmes erfolgt

```
umaxform infile outfile .
```

Sollten im Eingabefile mehr als zwei Datenspalten sein, so werden alle Spalten ab Spalte 3 ignoriert.

A.5 Programm zero.c

Mit zero.c läßt sich die erste Spalte eines Datenfiles in Nullage bringen, d. h. der erste Wert in Spalte 1 wird als Offset von allen Werten der Spalte 1 subtrahiert.
Die wird z. B. benötigt, wenn man verschiedene Datenfiles in denen Zuglängen unterschiedlich gehandhabt werden (Zugspitze, Zugmitte ...) miteinander vergleichen möchte.
Das Programm wird

```
zero infile
```

gerufen, die Ausgabe erscheint dann im File `infile.zero`.
Auch hier werden Kommentarzeilen mit einem führenden # kopiert.

A.6 Programm fzr2d-outconv.c

Die Ausgaben von Zeit-, Orts- und Wegdaten von fzr2d erfolgen nach `stdout` und können natürlich mit der Ausgabeumlenkung

```
fzr2d eingabefile parameterfile > aus.txt
```

in ein File `aus.txt` umgelenkt werden.
Mit Hilfe von diesem File lassen sich nun mit dem Programm fzr2d-outconv Gnuplot-Datenfiles zum Plotten von $t(s)$, $v(s)$- Verläufen erstellen.
Dem Programnm wird als Parameter der Name des fzr2d-Ausgabefiles und die Zuganzahl übergeben also würde z. B.:

```
fzr2d-outconv aus.txt 2
```

der korrekte Aufruf für eine Fahrzeitrechnung mit 2 Zügen sein.
Es ist günstig, das Programm in ein Verzeichnis zu kopieren, welches im Suchpfad steht, da man es in verschiedenen Ausgabeverzeichnissen verwenden wird.
Mögliche Plätze wären `/usr/local/bin` oder `$HOME/bin` .

A.7 Programm gen-hintereinander.c

Dieses Programm dient zur Erstellung von Eingabefiles zu Laufwegen und Sicherungsblöcken für die Hintereinanderfahrt zweier Züge auf der gleichen Strecke.
Im Quelltext ist als Konstante die Anzahl der Sicherungsblöcke anzugeben, es werden das Sicherungsblock-File und zwei Laufweg-Files für die beiden Züge generiert.
Die Daten der Strecke werden dabei per Zufall erzeugt, durch Änderung der Parameter in der enthaltenen Funktion `scramble_data` sind andere Verteilungen erzeugbar.

A.8 Programm acc.c

Mit Hilfe des Programmes acc.c lassen sich aus $v(t)$ Sampledaten Energiesummen berechnen.
Die Werte für v und t müssen dabei in Meter pro Sekunde und Sekunden angegeben sein. Dies läßt sich durch Setzen des entsprechenden Makros in fzr2d-outconv erreichen.
Der Aufruf erfolgt durch

```
acc v_t_datenfile
```

Die numerisch approximierten Werte der Beschleunigung sowie die Energiesumme erscheinen dann im File `at.dat`.

A.9 Durchführung der Testrechnungen zur Ermittlung der optimalen Wartezeit

Um die Arbeit bei der Testrechnung zur Ermittlung der optimalen Wartezeit des hinteren Zuges im Beispiel zweier Züge, die hintereinander fahren zu ermitteln, existieren im Verzeichnis fzr2d-test-warten einige Scripte und Hilfprogramme.

A.9.1 Ermitteln von Fahrzeiten bei verschiedenen Startzeiten

Zum Ermitteln von Ankunfts- und Fahrzeiten bei verschiedenen Startzeiten des hinteren Zuges wird im Programm do-simu.c der Spielraum und die Schrittweite der Wartezeit angegeben. Diese Programm ruft dann entsprechend oft die Fahrzeitrechnung und protokolliert die Ausgaben in `simu-aus.txt`.
Mittels des Programmes auswerten.c läßt sich dieses File dann automatisch auswerten um daraus das Gnuplot-Datenfile `warteplot.dat` zu erstellen.
In diesem stehen dann in 3 Spalten die Werte der Wartezeit, Ankunftszeit und Fahrzeit des hinteren Zuges.

A.9.2 Ermitteln der Energiesummen

Ausgangspunkt ist das Programm do-acalc.c. Dort ist das Intervall und die Schrittweite der zu variierenden Wartezeit anzugeben. Diese Programm ruft dann entsprechend die Fahrzeitrechnung und die Auswertungsprogramme auf und erstellt daraus das Gnuplot-Datenfile `energie.dat`.
Dabei ist eine Variante der Fahrzeitrechnung zu verwenden, die Simulationsschritte ausgibt.

A.10 Rechnen von Testreihen zu Laufzeitvergleichen

Um das gleiche Beispiel mit unterschiedlichen Simulationsschrittwiten zu rechnen gibt es die Tools im Verzeichnis fzr2d-laufzeit.

Das Programm `do-laufzeittest.c` erzeugt dafür, entsprechend den Vorgaben im Quelltext fortlaufend Parameterfiles mit den entsprechenden Schrittweiten, rechnet das Beispiel und ermittelt aus den Ausgaben die errechneten Ankunftszeiten der beiden Züge sowie die Rechenzeiten des Programms.
Sämtliche Werte werden dabei in eine Latex-Tabelle im File `fzr2d-laufzeittest.tex` geschrieben und können so in anderen Dokumenten weiterverarbeitet werden.

Anhang B

Unterschiede des zweiten Fzr1D-Programmes zum Standardprogramm

B.1 Ursache zur Entwicklung der zweiten Version

Das ursprüngliche Programm zur Fahrzeitrechnung entstand in der TLC und nutzt sehr viele C++ Optionen, die sich nicht ohne weiteres mit jedem Compiler compilieren lassen. Außerdem ist es dort in eine Testumgebung eingebaut.
Aus diesen Gründen ist diese zweite Version entstanden, die im logischen Aufbau zu dem anderen Programm identisch ist, sich aber mit jedem C++ Compiler übersetzen läßt und mit ASCII- Eingabefiles arbeitet.
Im weiteren eine kurze Beschreibung der Unterschiede in der Handhabung, Compilation zu dem Standard-Programm.

B.2 Compilation

Ausgangspunkt ist das Makefile, dort sind folgende Makros definierbar:

- FZR_TRACE_LOGIK ... Erzeugen einer Version mit sehr vielen Ausgaben
- ZEITMESSUNG ... Zeitmessung in den Programmteilen

Als Standardcompiler ist der Gnu-Compiler g++ eingestellt. Für andere Compiler ist die Variable CCOM im Makefile entsprechend zu setzen.
Compiliert wird das Programm durch Aufruf von make.

B.3 Ausführen

Nach erfolgreicher Compilation ist das Programm fzr entstanden. Wird es ohne Parametzer gerufen erfolgt ein usage-Hinweis und Abbruch. Der korrekte Ruf des Programmes erfolgt durch

```
./fzr fahrzeugfile streckenfile
```

Es erfolgen dann beim Lauf des Programmes je nach Version (FZR_TRACE_LOGIK, ZEITMESSUNG) entsprechende Ausgaben nach stdout und ins Ausgabeverzeichnis ./aus. Unter anderem entstehen in ./aus Gnuplot-Datenfiles, die mit dem Script plotall durch gnuplot dargestellt werden können.

```
cp plotall ./aus
cd ./aus
gnuplot plotall
```

Ausserdem ist es möglich, in `plotall` die Variable `PLOTPS` auf 1 zu setzen, dann erfolgt die Ausgabe in Postscript-Dateien anstatt auf den Monitor. Im weiteren eine Beschreibung der beiden Eingabefiles. In diesen Eingabefiles sind Kommentarzeilen durch ein # in der ersten Spalte einer Zeile gekennzeichnet, Leerzeilen sind nicht erlaubt.

B.4 Das Fahrzeugfile

Im Fahrzeugfile sind die fahrzeugbezogenen Daten des Zuges wie Masse, Reibungskäfte und geschwindigkeitsabhängige maximale Antriebskraft festzulegen.
Die Masse wird in Kilogramm angegeben und ist mit dem entsprechenden Zuschlag für die Erfassung der Rotationsenergie zu versehen.
Die geschwindigkeitsabhängigen Reibungskräfte werden durch drei Koeffizienten k_0, k_1 und k_2 in der folgenden Approximation

$$u_{reib}(v) = k_0 + k_1 v + k_2 v^2 \tag{B.1}$$

festgelegt. Dann folgen die gesampelten Werte für die maximale Antriebskraft in Abhängigkeit von der Geschwindigkeit in den Stützstellen 0, 0.1, 0.2 $\cdots$ m/s.
Hier ein Beispiel eines Fahrzeugfiles:

```
#
#  Beispiel eines Fahrzeugdaten-Files
#
#  Masse mit Rotationszuschlag in kg
  245000.0
# Reibungsterme fuer Reibungskraft u_reib=k_0+k_1*v+k_2*v^2
  10000.0 0.0 10.3
#
# Approximation von umax(v)
#      v [m/s]     umax(v) [N]
#
       0.000000    246084.000000
       0.100000    245272.000000
       0.200000    244462.000000
```

```
       0.300000    243656.000000
        ...           ...
# Ende des Files
```

B.5 Das Streckenfile

Im Streckenfile sind die wegbezogenen Daten der Zugfahrt festgelegt.
Dies sind zuerst die Anzahl der Wegabschnitte und dann für alle Wegabschnitte die Länge, Neigung, zulässige Maximalgeschwindigkeit, zulässige maximale Bremsverzögerung, Haltezeit und Name des Abschnittes.
Bei einer Haltezeit größer Null sei vereinbart, daß die Länge dieses Abschnittes gleich Null sei.
Auch hier ein Beispiel:

```
#
#  Beispiel eines Streckfiles
#
#  Anzahl der Wegabschnitte
  347
#
# Daten der Wegabschnitte
#
# laenge [m] neigung  vmax[km/h] amin[m/s^2] haltezeit[s] name
  148.600000 0.001040 100.000000 -0.700000     0.000000   Abschn1
    0.000000 0.001040   0.000000 -0.700000   600.000000   Abschn2
   20.000000 0.001040  90.000000 -0.700000     0.000000   Abschn3

    ...         ...     ...         ...           ...          ...

  159.000000 0.001760  90.000000 -0.700000    0.000000    Abschn347
# Ende des Files
```

Anhang C

Tabellen zum analytisch berechenbaren Testfall von fzr1d

Es folgen einige Tabellen mit Testrechnungen zum analytischen Beispiel der eindimensionalen Fahrzeitrechnung.
Dieses wurde bereits auf den Seiten 50 ff. bechrieben.
Es ist dabei die Abhängigkeit des Aufwandes und der Genauigkeit von Grade der Interpolation in den beiden Beispielen Intervallende und Aufsprungpunkt dargestellt.
Dabei bezeichnen die jeweiligen Spalten in den folgenden beiden Tabellengruppen die folgenden Größen

Grad Grad des Interplationspolynomes

eps_v Genauigkeitsschranke zur Schrittweitensteuerung

DGL-Steps Anzahl der durchgeführten Lösungsschritte des Differentialgleichungslösers

Akzeptiert Anzahl der durch die Schrittweitensteuerung akzeptierten Lösungsschritte

dtmin Kleinste in der Rechnung jemals verwendete Schrittweite

dtmax Größte in der Rechnung jemals verwendete Schrittweite

Zeit Errechnete Zeit

Fehler-Zeit Fehler in der berechneten Zeit (ermittelt durch Vergleich mit der analytischen Lösung)

Ort Errechneter Ort

Fehler-Ort Fehler in dem berechneten Ort (ermittelt durch Vergleich mit der analytischen Lösung)

Geschwindigkeit Errechnete Geschwindigkeit

Fehler-Geschwindigkeit Fehler in der berechneten Geschwindigkeit (ermittelt durch Vergleich mit der analytischen Lösung)

C.1 Variante zum Erreichen des Intervallendes

Grad	eps_v	DGL-Steps	Akzeptiert	dtmin	dtmax
1	1.0e+02	20	5	0.0100000000	244.6065496592
1	1.0e+01	12	4	0.0100000000	340.3027835253
1	1.0e+00	15	5	0.0100000000	433.3441767963
1	1.0e-01	18	6	0.0100000000	390.1204999382
1	1.0e-02	21	7	0.0100000000	248.5106336678
1	1.0e-03	27	9	0.0100000000	229.1393031041
1	1.0e-04	36	12	0.0100000000	207.3708310362
1	1.0e-05	48	16	0.0100000000	151.3730316434
1	1.0e-06	66	22	0.0100000000	105.4465220652
1	1.0e-07	93	31	0.0100000000	69.0189074651
1	1.0e-08	135	45	0.0100000000	44.5321017249
1	1.0e-09	201	67	0.0100000000	28.5247772133
1	1.0e-10	306	102	0.0100000000	18.3860113822
1	1.0e-11	471	157	0.0100000000	11.6503975645
1	1.0e-12	735	245	0.0100000000	7.4835637337
1	1.0e-13	1152	384	0.0100000000	4.7549845446
1	1.0e-14	1812	604	0.0100000000	3.0028494911

Tabelle C.1: Aufwand im Fall Intervallende mit Polynomgrad 1

Grad	eps_v	DGL-Steps	Akzeptiert	dtmin	dtmax
2	1.0e+02	21	5	0.0100000000	244.6065496592
2	1.0e+01	13	4	0.0100000000	340.3027835253
2	1.0e+00	16	5	0.0100000000	433.3441767963
2	1.0e-01	19	6	0.0100000000	390.1204999382
2	1.0e-02	22	7	0.0100000000	248.5106336678
2	1.0e-03	28	9	0.0100000000	229.1393031041
2	1.0e-04	37	12	0.0100000000	207.3708310362
2	1.0e-05	49	16	0.0100000000	151.3730316434
2	1.0e-06	67	22	0.0100000000	105.4465220652
2	1.0e-07	94	31	0.0100000000	69.0189074651
2	1.0e-08	136	45	0.0100000000	44.5321017249
2	1.0e-09	202	67	0.0100000000	28.5247772133
2	1.0e-10	307	102	0.0100000000	18.3860113822
2	1.0e-11	472	157	0.0100000000	11.6503975645
2	1.0e-12	736	245	0.0100000000	7.4835637337
2	1.0e-13	1153	384	0.0100000000	4.7549845446
2	1.0e-14	1813	604	0.0100000000	3.0028494911

Tabelle C.2: Aufwand im Fall Intervallende mit Polynomgrad 2

Grad	eps_v	DGL-Steps	Akzeptiert	dtmin	dtmax
3	1.0e+02	22	5	0.0100000000	244.6065496592
3	1.0e+01	14	4	0.0100000000	340.3027835253
3	1.0e+00	17	5	0.0100000000	433.3441767963
3	1.0e-01	20	6	0.0100000000	390.1204999382
3	1.0e-02	23	7	0.0100000000	248.5106336678
3	1.0e-03	29	9	0.0100000000	229.1393031041
3	1.0e-04	38	12	0.0100000000	207.3708310362
3	1.0e-05	50	16	0.0100000000	151.3730316434
3	1.0e-06	68	22	0.0100000000	105.4465220652
3	1.0e-07	95	31	0.0100000000	69.0189074651
3	1.0e-08	137	45	0.0100000000	44.5321017249
3	1.0e-09	203	67	0.0100000000	28.5247772133
3	1.0e-10	308	102	0.0100000000	18.3860113822
3	1.0e-11	473	157	0.0100000000	11.6503975645
3	1.0e-12	737	245	0.0100000000	7.4835637337
3	1.0e-13	1154	384	0.0100000000	4.7549845446
3	1.0e-14	1814	604	0.0100000000	3.0028494911

Tabelle C.3: Aufwand im Fall Intervallende mit Polynomgrad 3

Grad	eps_v	DGL-Steps	Akzeptiert	dtmin	dtmax
4	1.0e+02	23	5	0.0100000000	244.6065496592
4	1.0e+01	15	4	0.0100000000	340.3027835253
4	1.0e+00	18	5	0.0100000000	433.3441767963
4	1.0e-01	21	6	0.0100000000	390.1204999382
4	1.0e-02	24	7	0.0100000000	248.5106336678
4	1.0e-03	30	9	0.0100000000	229.1393031041
4	1.0e-04	39	12	0.0100000000	207.3708310362
4	1.0e-05	51	16	0.0100000000	151.3730316434
4	1.0e-06	69	22	0.0100000000	105.4465220652
4	1.0e-07	96	31	0.0100000000	69.0189074651
4	1.0e-08	138	45	0.0100000000	44.5321017249
4	1.0e-09	204	67	0.0100000000	28.5247772133
4	1.0e-10	309	102	0.0100000000	18.3860113822
4	1.0e-11	474	157	0.0100000000	11.6503975645
4	1.0e-12	738	245	0.0100000000	7.4835637337
4	1.0e-13	1155	384	0.0100000000	4.7549845446
4	1.0e-14	1815	604	0.0100000000	3.0028494911

Tabelle C.4: Aufwand im Fall Intervallende mit Polynomgrad 4

Grad	eps_v	DGL-Steps	Akzeptiert	dtmin	dtmax
5	1.0e+02	24	5	0.0100000000	244.6065496592
5	1.0e+01	16	4	0.0100000000	340.3027835253
5	1.0e+00	19	5	0.0100000000	433.3441767963
5	1.0e-01	22	6	0.0100000000	390.1204999382
5	1.0e-02	25	7	0.0100000000	248.5106336678
5	1.0e-03	31	9	0.0100000000	229.1393031041
5	1.0e-04	40	12	0.0100000000	207.3708310362
5	1.0e-05	52	16	0.0100000000	151.3730316434
5	1.0e-06	70	22	0.0100000000	105.4465220652
5	1.0e-07	97	31	0.0100000000	69.0189074651
5	1.0e-08	139	45	0.0100000000	44.5321017249
5	1.0e-09	205	67	0.0100000000	28.5247772133
5	1.0e-10	310	102	0.0100000000	18.3860113822
5	1.0e-11	475	157	0.0100000000	11.6503975645
5	1.0e-12	739	245	0.0100000000	7.4835637337
5	1.0e-13	1156	384	0.0100000000	4.7549845446
5	1.0e-14	1816	604	0.0100000000	3.0028494911

Tabelle C.5: Aufwand im Fall Intervallende mit Polynomgrad 5

Grad	eps_v	DGL-Steps	Akzeptiert	dtmin	dtmax
6	1.0e+02	25	5	0.0100000000	244.6065496592
6	1.0e+01	17	4	0.0100000000	340.3027835253
6	1.0e+00	20	5	0.0100000000	433.3441767963
6	1.0e-01	23	6	0.0100000000	390.1204999382
6	1.0e-02	26	7	0.0100000000	248.5106336678
6	1.0e-03	32	9	0.0100000000	229.1393031041
6	1.0e-04	41	12	0.0100000000	207.3708310362
6	1.0e-05	53	16	0.0100000000	151.3730316434
6	1.0e-06	71	22	0.0100000000	105.4465220652
6	1.0e-07	98	31	0.0100000000	69.0189074651
6	1.0e-08	140	45	0.0100000000	44.5321017249
6	1.0e-09	206	67	0.0100000000	28.5247772133
6	1.0e-10	311	102	0.0100000000	18.3860113822
6	1.0e-11	476	157	0.0100000000	11.6503975645
6	1.0e-12	740	245	0.0100000000	7.4835637337
6	1.0e-13	1157	384	0.0100000000	4.7549845446
6	1.0e-14	1817	604	0.0100000000	3.0028494911

Tabelle C.6: Aufwand im Fall Intervallende mit Polynomgrad 6

Grad	eps_v	DGL-Steps	Akzeptiert	dtmin	dtmax
7	1.0e+02	26	5	0.0100000000	244.6065496592
7	1.0e+01	18	4	0.0100000000	340.3027835253
7	1.0e+00	21	5	0.0100000000	433.3441767963
7	1.0e-01	24	6	0.0100000000	390.1204999382
7	1.0e-02	27	7	0.0100000000	248.5106336678
7	1.0e-03	33	9	0.0100000000	229.1393031041
7	1.0e-04	42	12	0.0100000000	207.3708310362
7	1.0e-05	54	16	0.0100000000	151.3730316434
7	1.0e-06	72	22	0.0100000000	105.4465220652
7	1.0e-07	99	31	0.0100000000	69.0189074651
7	1.0e-08	141	45	0.0100000000	44.5321017249
7	1.0e-09	207	67	0.0100000000	28.5247772133
7	1.0e-10	312	102	0.0100000000	18.3860113822
7	1.0e-11	477	157	0.0100000000	11.6503975645
7	1.0e-12	741	245	0.0100000000	7.4835637337
7	1.0e-13	1158	384	0.0100000000	4.7549845446
7	1.0e-14	1818	604	0.0100000000	3.0028494911

Tabelle C.7: Aufwand im Fall Intervallende mit Polynomgrad 7

Grad	eps_v	DGL-Steps	Akzeptiert	dtmin	dtmax
8	1.0e+02	27	5	0.0100000000	244.6065496592
8	1.0e+01	19	4	0.0100000000	340.3027835253
8	1.0e+00	22	5	0.0100000000	433.3441767963
8	1.0e-01	25	6	0.0100000000	390.1204999382
8	1.0e-02	28	7	0.0100000000	248.5106336678
8	1.0e-03	34	9	0.0100000000	229.1393031041
8	1.0e-04	43	12	0.0100000000	207.3708310362
8	1.0e-05	55	16	0.0100000000	151.3730316434
8	1.0e-06	73	22	0.0100000000	105.4465220652
8	1.0e-07	100	31	0.0100000000	69.0189074651
8	1.0e-08	142	45	0.0100000000	44.5321017249
8	1.0e-09	208	67	0.0100000000	28.5247772133
8	1.0e-10	313	102	0.0100000000	18.3860113822
8	1.0e-11	478	157	0.0100000000	11.6503975645
8	1.0e-12	742	245	0.0100000000	7.4835637337
8	1.0e-13	1159	384	0.0100000000	4.7549845446
8	1.0e-14	1819	604	0.0100000000	3.0028494911

Tabelle C.8: Aufwand im Fall Intervallende mit Polynomgrad 8

Grad	eps_v	DGL-Steps	Akzeptiert	dtmin	dtmax
9	1.0e+02	28	5	0.0100000000	244.6065496592
9	1.0e+01	20	4	0.0100000000	340.3027835253
9	1.0e+00	23	5	0.0100000000	433.3441767963
9	1.0e-01	26	6	0.0100000000	390.1204999382
9	1.0e-02	29	7	0.0100000000	248.5106336678
9	1.0e-03	35	9	0.0100000000	229.1393031041
9	1.0e-04	44	12	0.0100000000	207.3708310362
9	1.0e-05	56	16	0.0100000000	151.3730316434
9	1.0e-06	74	22	0.0100000000	105.4465220652
9	1.0e-07	101	31	0.0100000000	69.0189074651
9	1.0e-08	143	45	0.0100000000	44.5321017249
9	1.0e-09	209	67	0.0100000000	28.5247772133
9	1.0e-10	314	102	0.0100000000	18.3860113822
9	1.0e-11	479	157	0.0100000000	11.6503975645
9	1.0e-12	743	245	0.0100000000	7.4835637337
9	1.0e-13	1160	384	0.0100000000	4.7549845446
9	1.0e-14	1820	604	0.0100000000	3.0028494911

Tabelle C.9: Aufwand im Fall Intervallende mit Polynomgrad 9

Grad	eps_v	Zeit	Fehler-Zeit	Geschwindigkeit	Fehler-Geschw.
1	1.0e+02	389.4204018496	-9.5795981504	20.0272809405	0.0272809405
1	1.0e+01	385.2575319486	-13.7424680514	19.7216735075	-0.2783264925
1	1.0e+00	364.2248711520	-34.7751288480	18.3093223790	-1.6906776210
1	1.0e-01	373.3683623896	-25.6316376104	18.7334737646	-1.2665262354
1	1.0e-02	392.5025079864	-6.4974920136	19.6651295612	-0.3348704388
1	1.0e-03	390.5942645605	-8.4057354395	19.5701726853	-0.4298273147
1	1.0e-04	394.3694721633	-4.6305278367	19.7778371176	-0.2221628824
1	1.0e-05	397.4594803091	-1.5405196909	19.9264233375	-0.0735766625
1	1.0e-06	398.9175839234	-0.0824160766	19.9961642409	-0.0038357591
1	1.0e-07	398.8852861401	-0.1147138599	19.9944285597	-0.0055714403
1	1.0e-08	398.8619247665	-0.1380752335	19.9931886149	-0.0068113851
1	1.0e-09	398.8974261572	-0.1025738428	19.9948955208	-0.0051044792
1	1.0e-10	398.9595323450	-0.0404676550	19.9979823599	-0.0020176401
1	1.0e-11	398.9794677365	-0.0205322635	19.9989708751	-0.0010291249
1	1.0e-12	398.9941744626	-0.0058255374	19.9997076508	-0.0002923492
1	1.0e-13	398.9995402040	-0.0004597960	19.9999754604	-0.0000245396
1	1.0e-14	398.9998468971	-0.0001531029	19.9999907608	-0.0000092392

Tabelle C.10: Fehler im Fall Intervallende mit Polynomgrad 1

Grad	eps_v	Zeit	Fehler-Zeit	Geschwindigkeit	Fehler-Geschw.
2	1.0e+02	389.8847138871	-9.1152861129	20.0486285806	0.0486285806
2	1.0e+01	394.4976721657	-4.5023278343	20.3397104782	0.3397104782
2	1.0e+00	397.8314929690	-1.1685070310	20.0970258889	0.0970258889
2	1.0e-01	398.0446560974	-0.9553439026	20.0022458382	0.0022458382
2	1.0e-02	399.7818861249	0.7818861249	20.0859323943	0.0859323943
2	1.0e-03	399.3173422427	0.3173422427	20.0292043398	0.0292043398
2	1.0e-04	398.6098876964	-0.3901123036	19.9789323760	-0.0210676240
2	1.0e-05	398.8662480540	-0.1337519460	19.9926641212	-0.0073358788
2	1.0e-06	398.9913225156	-0.0086774844	19.9996425133	-0.0003574867
2	1.0e-07	398.9938948309	-0.0061051691	19.9996619936	-0.0003380064
2	1.0e-08	398.9962957651	-0.0037042349	19.9997769274	-0.0002230726
2	1.0e-09	398.9989685025	-0.0010314975	19.9999353649	-0.0000646351
2	1.0e-10	398.9997203208	-0.0002796792	19.9999810247	-0.0000189753
2	1.0e-11	399.0000517909	0.0000517909	20.0000014847	0.0000014847
2	1.0e-12	398.9999949890	-0.0000050110	19.9999979031	-0.0000020969
2	1.0e-13	399.0000141248	0.0000141248	19.9999990914	-0.0000009086
2	1.0e-14	399.0000154862	0.0000154862	19.9999991754	-0.0000008246

Tabelle C.11: Fehler im Fall Intervallende mit Polynomgrad 2

Grad	eps_v	Zeit	Fehler-Zeit	Geschwindigkeit	Fehler-Geschw.
3	1.0e+02	389.9326171960	-9.0673828040	20.0513973093	0.0513973093
3	1.0e+01	389.9380773710	-9.0619226290	19.9338972980	-0.0661027020
3	1.0e+00	397.3555313691	-1.6444686309	20.0714377209	0.0714377209
3	1.0e-01	398.4978125745	-0.5021874255	20.0359168651	0.0359168651
3	1.0e-02	398.3687660517	-0.6312339483	19.9825202434	-0.0174797566
3	1.0e-03	398.9040387411	-0.0959612589	20.0021965456	0.0021965456
3	1.0e-04	398.9411895658	-0.0588104342	19.9988067223	-0.0011932777
3	1.0e-05	398.9789221403	-0.0210778597	19.9993365103	-0.0006634897
3	1.0e-06	398.9968316448	-0.0031683552	19.9999683520	-0.0000316480
3	1.0e-07	398.9991429325	-0.0008570675	19.9999786907	-0.0000213093
3	1.0e-08	398.9998068113	-0.0001931887	19.9999923921	-0.0000076079
3	1.0e-09	398.9999902533	-0.0000097467	19.9999988347	-0.0000011653
3	1.0e-10	399.0000108919	0.0000108919	19.9999991012	-0.0000008988
3	1.0e-11	399.0000154771	0.0000154771	19.9999992145	-0.0000007855
3	1.0e-12	399.0000157186	0.0000157186	19.9999991955	-0.0000008045
3	1.0e-13	399.0000158602	0.0000158602	19.9999991996	-0.0000008004
3	1.0e-14	399.0000158803	0.0000158803	19.9999992000	-0.0000008000

Tabelle C.12: Fehler im Fall Intervallende mit Polynomgrad 3

Grad	eps_v	Zeit	Fehler-Zeit	Geschwindigkeit	Fehler-Geschw.
4	1.0e+02	389.9382682366	-9.0617317634	20.0517685437	0.0517685437
4	1.0e+01	392.6465501896	-6.3534498104	20.2157881764	0.2157881764
4	1.0e+00	396.5514576861	-2.4485423139	20.0072998531	0.0072998531
4	1.0e-01	398.0902073672	-0.9097926328	20.0052962151	0.0052962151
4	1.0e-02	398.6574229400	-0.3425770600	20.0064460791	0.0064460791
4	1.0e-03	398.8738680950	-0.1261319050	19.9998891543	-0.0001108457
4	1.0e-04	398.9606650818	-0.0393349182	20.0001392318	0.0001392318
4	1.0e-05	398.9882958043	-0.0117041957	19.9999678323	-0.0000321677
4	1.0e-06	398.9973089892	-0.0026910108	20.0000004839	0.0000004839
4	1.0e-07	398.9994317269	-0.0005682731	19.9999985488	-0.0000014512
4	1.0e-08	398.9999025193	-0.0000974807	19.9999990920	-0.0000009080
4	1.0e-09	398.9999956340	-0.0000043660	19.9999992169	-0.0000007831
4	1.0e-10	399.0000123042	0.0000123042	19.9999992016	-0.0000007984
4	1.0e-11	399.0000152766	0.0000152766	19.9999992001	-0.0000007999
4	1.0e-12	399.0000157829	0.0000157829	19.9999992001	-0.0000007999
4	1.0e-13	399.0000158675	0.0000158675	19.9999992001	-0.0000007999
4	1.0e-14	399.0000158814	0.0000158814	19.9999992001	-0.0000007999

Tabelle C.13: Fehler im Fall Intervallende mit Polynomgrad 4

Grad	eps_v	Zeit	Fehler-Zeit	Geschwindigkeit	Fehler-Geschw.
5	1.0e+02	389.9389734960	-9.0610265040	20.0518195715	0.0518195715
5	1.0e+01	390.8881083502	-8.1118916498	20.0140359606	0.0140359606
5	1.0e+00	396.6155281591	-2.3844718409	20.0126273580	0.0126273580
5	1.0e-01	398.0723788694	-0.9276211306	20.0037437781	0.0037437781
5	1.0e-02	398.6018644503	-0.3981355497	20.0014206937	0.0014206937
5	1.0e-03	398.8815470173	-0.1184529827	20.0005207373	0.0005207373
5	1.0e-04	398.9608230747	-0.0391769253	20.0001522053	0.0001522053
5	1.0e-05	398.9890422125	-0.0109577875	20.0000232466	0.0000232466
5	1.0e-06	398.9973532283	-0.0026467717	20.0000037710	0.0000037710
5	1.0e-07	398.9994485031	-0.0005514969	19.9999998244	-0.0000001756
5	1.0e-08	398.9999050357	-0.0000949643	19.9999992870	-0.0000007130
5	1.0e-09	398.9999955629	-0.0000044371	19.9999992113	-0.0000007887
5	1.0e-10	399.0000123013	0.0000123013	19.9999992014	-0.0000007986
5	1.0e-11	399.0000152781	0.0000152781	19.9999992002	-0.0000007998
5	1.0e-12	399.0000157830	0.0000157830	19.9999992001	-0.0000007999
5	1.0e-13	399.0000158675	0.0000158675	19.9999992001	-0.0000007999
5	1.0e-14	399.0000158814	0.0000158814	19.9999992001	-0.0000007999

Tabelle C.14: Fehler im Fall Intervallende mit Polynomgrad 5

Grad	eps_v	Zeit	Fehler-Zeit	Geschwindigkeit	Fehler-Geschw.
6	1.0e+02	389.9390641969	-9.0609358031	20.0518266898	0.0518266898
6	1.0e+01	392.0812704929	-6.9187295071	20.1608198692	0.1608198692
6	1.0e+00	396.6565967445	-2.3434032555	20.0164654888	0.0164654888
6	1.0e-01	398.0875471671	-0.9124528329	20.0050762087	0.0050762087
6	1.0e-02	398.6109095590	-0.3890904410	20.0022979534	0.0022979534
6	1.0e-03	398.8812564220	-0.1187435780	20.0004955920	0.0004955920
6	1.0e-04	398.9606664534	-0.0393335466	20.0001396568	0.0001396568
6	1.0e-05	398.9890958840	-0.0109041160	20.0000275739	0.0000275739
6	1.0e-06	398.9973574923	-0.0026425077	20.0000041155	0.0000041155
6	1.0e-07	398.9994495056	-0.0005504944	19.9999999074	-0.0000000926
6	1.0e-08	398.9999050963	-0.0000949037	19.9999992921	-0.0000007079
6	1.0e-09	398.9999955605	-0.0000044395	19.9999992111	-0.0000007889
6	1.0e-10	399.0000123011	0.0000123011	19.9999992014	-0.0000007986
6	1.0e-11	399.0000152781	0.0000152781	19.9999992002	-0.0000007998
6	1.0e-12	399.0000157830	0.0000157830	19.9999992001	-0.0000007999
6	1.0e-13	399.0000158675	0.0000158675	19.9999992001	-0.0000007999
6	1.0e-14	399.0000158814	0.0000158814	19.9999992001	-0.0000007999

Tabelle C.15: Fehler im Fall Intervallende mit Polynomgrad 6

Grad	eps_v	Zeit	Fehler-Zeit	Geschwindigkeit	Fehler-Geschw.
7	1.0e+02	389.9390760712	-9.0609239288	20.0518276910	0.0518276910
7	1.0e+01	391.2533267105	-7.7466732895	20.0531681884	0.0531681884
7	1.0e+00	396.6495280980	-2.3504719020	20.0157828087	0.0157828087
7	1.0e-01	398.0884398721	-0.9115601279	20.0051631853	0.0051631853
7	1.0e-02	398.6100296785	-0.3899703215	20.0022079534	0.0022079534
7	1.0e-03	398.8811500343	-0.1188499657	20.0004854412	0.0004854412
7	1.0e-04	398.9606413072	-0.0393586928	20.0001374837	0.0001374837
7	1.0e-05	398.9890989973	-0.0109010027	20.0000278442	0.0000278442
7	1.0e-06	398.9973579141	-0.0026420859	20.0000041522	0.0000041522
7	1.0e-07	398.9994495665	-0.0005504335	19.9999999128	-0.0000000872
7	1.0e-08	398.9999050975	-0.0000949025	19.9999992922	-0.0000007078
7	1.0e-09	398.9999955605	-0.0000044395	19.9999992111	-0.0000007889
7	1.0e-10	399.0000123011	0.0000123011	19.9999992014	-0.0000007986
7	1.0e-11	399.0000152781	0.0000152781	19.9999992002	-0.0000007998
7	1.0e-12	399.0000157830	0.0000157830	19.9999992001	-0.0000007999
7	1.0e-13	399.0000158675	0.0000158675	19.9999992001	-0.0000007999
7	1.0e-14	399.0000158814	0.0000158814	19.9999992001	-0.0000007999

Tabelle C.16: Fehler im Fall Intervallende mit Polynomgrad 7

Grad	eps_v	Zeit	Fehler-Zeit	Geschwindigkeit	Fehler-Geschw.
8	1.0e+02	389.9390776427	-9.0609223573	20.0518278323	0.0518278323
8	1.0e+01	391.8339612500	-7.1660387500	20.1322927264	0.1322927264
8	1.0e+00	396.6471027346	-2.3528972654	20.0155296898	0.0155296898
8	1.0e-01	398.0877204359	-0.9122795641	20.0050924810	0.0050924810
8	1.0e-02	398.6098842980	-0.3901157020	20.0021916396	0.0021916396
8	1.0e-03	398.8811673524	-0.1188326476	20.0004871787	0.0004871787
8	1.0e-04	398.9606392170	-0.0393607830	20.0001372905	0.0001372905
8	1.0e-05	398.9890990803	-0.0109009197	20.0000278520	0.0000278520
8	1.0e-06	398.9973579566	-0.0026420434	20.0000041561	0.0000041561
8	1.0e-07	398.9994495702	-0.0005504298	19.9999999132	-0.0000000868
8	1.0e-08	398.9999050975	-0.0000949025	19.9999992923	-0.0000007077
8	1.0e-09	398.9999955605	-0.0000044395	19.9999992111	-0.0000007889
8	1.0e-10	399.0000123011	0.0000123011	19.9999992014	-0.0000007986
8	1.0e-11	399.0000152781	0.0000152781	19.9999992002	-0.0000007998
8	1.0e-12	399.0000157830	0.0000157830	19.9999992001	-0.0000007999
8	1.0e-13	399.0000158675	0.0000158675	19.9999992001	-0.0000007999
8	1.0e-14	399.0000158814	0.0000158814	19.9999992001	-0.0000007999

Tabelle C.17: Fehler im Fall Intervallende mit Polynomgrad 8

Grad	eps_v	Zeit	Fehler-Zeit	Geschwindigkeit	Fehler-Geschw.
9	1.0e+02	389.9390778521	-9.0609221479	20.0518278523	0.0518278523
9	1.0e+01	391.4254547224	-7.5745452776	20.0742682537	0.0742682537
9	1.0e+00	396.6478373160	-2.3521626840	20.0156084128	0.0156084128
9	1.0e-01	398.0876701538	-0.9123298462	20.0050871061	0.0050871061
9	1.0e-02	398.6100071184	-0.3899928816	20.0022059209	0.0022059209
9	1.0e-03	398.8811676301	-0.1188323699	20.0004872102	0.0004872102
9	1.0e-04	398.9606391954	-0.0393608046	20.0001372882	0.0001372882
9	1.0e-05	398.9890990673	-0.0109009327	20.0000278507	0.0000278507
9	1.0e-06	398.9973579610	-0.0026420390	20.0000041566	0.0000041566
9	1.0e-07	398.9994495705	-0.0005504295	19.9999999132	-0.0000000868
9	1.0e-08	398.9999050975	-0.0000949025	19.9999992923	-0.0000007077
9	1.0e-09	398.9999955605	-0.0000044395	19.9999992111	-0.0000007889
9	1.0e-10	399.0000123011	0.0000123011	19.9999992014	-0.0000007986
9	1.0e-11	399.0000152781	0.0000152781	19.9999992002	-0.0000007998
9	1.0e-12	399.0000157830	0.0000157830	19.9999992001	-0.0000007999
9	1.0e-13	399.0000158675	0.0000158675	19.9999992001	-0.0000007999
9	1.0e-14	399.0000158814	0.0000158814	19.9999992001	-0.0000007999

Tabelle C.18: Fehler im Fall Intervallende mit Polynomgrad 9

C.2 Variante zum Finden eines Aufsprungpunktes

Grad	eps_v	DGL-Steps	Akzeptiert	dtmin	dtmax
1	1.0e2	12	4	0.0100000000	978.4261986369
1	1.0e1	16	5	0.0100000000	729.0919112733
1	1.0e0	19	6	0.0100000000	747.5310999180
1	1.0e-1	22	7	0.0100000000	516.8152649767
1	1.0e-2	24	8	0.0100000000	540.8745322205
1	1.0e-3	34	11	0.0100000000	411.9819849758
1	1.0e-4	42	14	0.0100000000	481.7617926568
1	1.0e-5	54	18	0.0100000000	277.4085047862
1	1.0e-6	75	25	0.0100000000	197.7512927572
1	1.0e-7	108	36	0.0100000000	140.0622433635
1	1.0e-8	156	52	0.0100000000	86.1084794028
1	1.0e-9	237	79	0.0100000000	59.5245709341
1	1.0e-10	360	120	0.0100000000	37.4935769010
1	1.0e-11	558	186	0.0100000000	24.2904375454
1	1.0e-12	870	290	0.0100000000	15.4634050613
1	1.0e-13	1365	455	0.0100000000	9.8349432218
1	1.0e-14	2148	716	0.0100000000	6.2262140465

Tabelle C.19: Aufwand im Fall Aufsprung mit Polynomgrad 1

Grad	eps_v	DGL-Steps	Akzeptiert	dtmin	dtmax
2	1.0e2	-1	-1	-1.0000000000	-1.0000000000
2	1.0e1	18	5	0.0100000000	729.0919112733
2	1.0e0	21	6	0.0100000000	747.5310999180
2	1.0e-1	24	7	0.0100000000	516.8152649767
2	1.0e-2	25	8	0.0100000000	540.8745322205
2	1.0e-3	36	11	0.0100000000	411.9819849758
2	1.0e-4	44	14	0.0100000000	481.7617926568
2	1.0e-5	56	18	0.0100000000	277.4085047862
2	1.0e-6	76	25	0.0100000000	197.7512927572
2	1.0e-7	110	36	0.0100000000	140.0622433635
2	1.0e-8	157	52	0.0100000000	86.1084794028
2	1.0e-9	238	79	0.0100000000	59.5245709341
2	1.0e-10	361	120	0.0100000000	37.4935769010
2	1.0e-11	559	186	0.0100000000	24.2904375454
2	1.0e-12	871	290	0.0100000000	15.4634050613
2	1.0e-13	1366	455	0.0100000000	9.8349432218
2	1.0e-14	2149	716	0.0100000000	6.2262140465

Tabelle C.20: Aufwand im Fall Aufsprung mit Polynomgrad 2

Grad	eps_v	DGL-Steps	Akzeptiert	dtmin	dtmax
3	1.0e2	-1	-1	-1.0000000000	-1.0000000000
3	1.0e1	20	5	0.0100000000	729.0919112733
3	1.0e0	23	6	0.0100000000	747.5310999180
3	1.0e-1	26	7	0.0100000000	516.8152649767
3	1.0e-2	26	8	0.0100000000	540.8745322205
3	1.0e-3	38	11	0.0100000000	411.9819849758
3	1.0e-4	46	14	0.0100000000	481.7617926568
3	1.0e-5	58	18	0.0100000000	277.4085047862
3	1.0e-6	77	25	0.0100000000	197.7512927572
3	1.0e-7	112	36	0.0100000000	140.0622433635
3	1.0e-8	158	52	0.0100000000	86.1084794028
3	1.0e-9	239	79	0.0100000000	59.5245709341
3	1.0e-10	362	120	0.0100000000	37.4935769010
3	1.0e-11	560	186	0.0100000000	24.2904375454
3	1.0e-12	872	290	0.0100000000	15.4634050613
3	1.0e-13	1367	455	0.0100000000	9.8349432218
3	1.0e-14	2150	716	0.0100000000	6.2262140465

Tabelle C.21: Aufwand im Fall Aufsprung mit Polynomgrad 3

Grad	eps_v	DGL-Steps	Akzeptiert	dtmin	dtmax
4	1.0e2	-1	-1	-1.0000000000	-1.0000000000
4	1.0e1	22	5	0.0100000000	729.0919112733
4	1.0e0	25	6	0.0100000000	747.5310999180
4	1.0e-1	28	7	0.0100000000	516.8152649767
4	1.0e-2	27	8	0.0100000000	540.8745322205
4	1.0e-3	40	11	0.0100000000	411.9819849758
4	1.0e-4	48	14	0.0100000000	481.7617926568
4	1.0e-5	60	18	0.0100000000	277.4085047862
4	1.0e-6	78	25	0.0100000000	197.7512927572
4	1.0e-7	114	36	0.0100000000	140.0622433635
4	1.0e-8	159	52	0.0100000000	86.1084794028
4	1.0e-9	240	79	0.0100000000	59.5245709341
4	1.0e-10	363	120	0.0100000000	37.4935769010
4	1.0e-11	561	186	0.0100000000	24.2904375454
4	1.0e-12	873	290	0.0100000000	15.4634050613
4	1.0e-13	1368	455	0.0100000000	9.8349432218
4	1.0e-14	2151	716	0.0100000000	6.2262140465

Tabelle C.22: Aufwand im Fall Aufsprung mit Polynomgrad 4

Grad	eps_v	DGL-Steps	Akzeptiert	dtmin	dtmax
5	1.0e2	-1	-1	-1.0000000000	-1.0000000000
5	1.0e1	24	5	0.0100000000	729.0919112733
5	1.0e0	27	6	0.0100000000	747.5310999180
5	1.0e-1	30	7	0.0100000000	516.8152649767
5	1.0e-2	28	8	0.0100000000	540.8745322205
5	1.0e-3	42	11	0.0100000000	411.9819849758
5	1.0e-4	50	14	0.0100000000	481.7617926568
5	1.0e-5	62	18	0.0100000000	277.4085047862
5	1.0e-6	79	25	0.0100000000	197.7512927572
5	1.0e-7	116	36	0.0100000000	140.0622433635
5	1.0e-8	160	52	0.0100000000	86.1084794028
5	1.0e-9	241	79	0.0100000000	59.5245709341
5	1.0e-10	364	120	0.0100000000	37.4935769010
5	1.0e-11	562	186	0.0100000000	24.2904375454
5	1.0e-12	874	290	0.0100000000	15.4634050613
5	1.0e-13	1369	455	0.0100000000	9.8349432218
5	1.0e-14	2152	716	0.0100000000	6.2262140465

Tabelle C.23: Aufwand im Fall Aufsprung mit Polynomgrad 5

Grad	eps_v	DGL-Steps	Akzeptiert	dtmin	dtmax
6	1.0e2	-1	-1	-1.0000000000	-1.0000000000
6	1.0e1	26	5	0.0100000000	729.0919112733
6	1.0e0	29	6	0.0100000000	747.5310999180
6	1.0e-1	32	7	0.0100000000	516.8152649767
6	1.0e-2	29	8	0.0100000000	540.8745322205
6	1.0e-3	44	11	0.0100000000	411.9819849758
6	1.0e-4	52	14	0.0100000000	481.7617926568
6	1.0e-5	64	18	0.0100000000	277.4085047862
6	1.0e-6	80	25	0.0100000000	197.7512927572
6	1.0e-7	118	36	0.0100000000	140.0622433635
6	1.0e-8	161	52	0.0100000000	86.1084794028
6	1.0e-9	242	79	0.0100000000	59.5245709341
6	1.0e-10	365	120	0.0100000000	37.4935769010
6	1.0e-11	563	186	0.0100000000	24.2904375454
6	1.0e-12	875	290	0.0100000000	15.4634050613
6	1.0e-13	1370	455	0.0100000000	9.8349432218
6	1.0e-14	2153	716	0.0100000000	6.2262140465

Tabelle C.24: Aufwand im Fall Aufsprung mit Polynomgrad 6

Grad	eps_v	DGL-Steps	Akzeptiert	dtmin	dtmax
7	1.0e2	-1	-1	-1.0000000000	-1.0000000000
7	1.0e1	28	5	0.0100000000	729.0919112733
7	1.0e0	31	6	0.0100000000	747.5310999180
7	1.0e-1	34	7	0.0100000000	516.8152649767
7	1.0e-2	30	8	0.0100000000	540.8745322205
7	1.0e-3	46	11	0.0100000000	411.9819849758
7	1.0e-4	54	14	0.0100000000	481.7617926568
7	1.0e-5	66	18	0.0100000000	277.4085047862
7	1.0e-6	81	25	0.0100000000	197.7512927572
7	1.0e-7	120	36	0.0100000000	140.0622433635
7	1.0e-8	162	52	0.0100000000	86.1084794028
7	1.0e-9	243	79	0.0100000000	59.5245709341
7	1.0e-10	366	120	0.0100000000	37.4935769010
7	1.0e-11	564	186	0.0100000000	24.2904375454
7	1.0e-12	876	290	0.0100000000	15.4634050613
7	1.0e-13	1371	455	0.0100000000	9.8349432218
7	1.0e-14	2154	716	0.0100000000	6.2262140465

Tabelle C.25: Aufwand im Fall Aufsprung mit Polynomgrad 7

Grad	eps_v	DGL-Steps	Akzeptiert	dtmin	dtmax
8	1.0e2	-1	-1	-1.0000000000	-1.0000000000
8	1.0e1	30	5	0.0100000000	729.0919112733
8	1.0e0	33	6	0.0100000000	747.5310999180
8	1.0e-1	36	7	0.0100000000	516.8152649767
8	1.0e-2	31	8	0.0100000000	540.8745322205
8	1.0e-3	48	11	0.0100000000	411.9819849758
8	1.0e-4	56	14	0.0100000000	481.7617926568
8	1.0e-5	68	18	0.0100000000	277.4085047862
8	1.0e-6	82	25	0.0100000000	197.7512927572
8	1.0e-7	122	36	0.0100000000	140.0622433635
8	1.0e-8	163	52	0.0100000000	86.1084794028
8	1.0e-9	244	79	0.0100000000	59.5245709341
8	1.0e-10	367	120	0.0100000000	37.4935769010
8	1.0e-11	565	186	0.0100000000	24.2904375454
8	1.0e-12	877	290	0.0100000000	15.4634050613
8	1.0e-13	1372	455	0.0100000000	9.8349432218
8	1.0e-14	2155	716	0.0100000000	6.2262140465

Tabelle C.26: Aufwand im Fall Aufsprung mit Polynomgrad 8

Grad	eps_v	DGL-Steps	Akzeptiert	dtmin	dtmax
9	1.0e2	-1	-1	-1.0000000000	-1.0000000000
9	1.0e1	32	5	0.0100000000	729.0919112733
9	1.0e0	35	6	0.0100000000	747.5310999180
9	1.0e-1	38	7	0.0100000000	516.8152649767
9	1.0e-2	32	8	0.0100000000	540.8745322205
9	1.0e-3	50	11	0.0100000000	411.9819849758
9	1.0e-4	58	14	0.0100000000	481.7617926568
9	1.0e-5	70	18	0.0100000000	277.4085047862
9	1.0e-6	83	25	0.0100000000	197.7512927572
9	1.0e-7	124	36	0.0100000000	140.0622433635
9	1.0e-8	164	52	0.0100000000	86.1084794028
9	1.0e-9	245	79	0.0100000000	59.5245709341
9	1.0e-10	368	120	0.0100000000	37.4935769010
9	1.0e-11	566	186	0.0100000000	24.2904375454
9	1.0e-12	878	290	0.0100000000	15.4634050613
9	1.0e-13	1373	455	0.0100000000	9.8349432218
9	1.0e-14	2156	716	0.0100000000	6.2262140465

Tabelle C.27: Aufwand im Fall Aufsprung mit Polynomgrad 9

Grad	eps_v	Zeit	Fehler-Zeit	Ort	Fehler-Ort
1	1.0e2	-1.0000000000	-900.0000000000	-1.0000000000	-18000.3333333333
1	1.0e1	922.3772109878	23.3772109878	20000.2803038626	2000.9469705293
1	1.0e0	929.7688226142	30.7688226142	20203.4528114011	2204.1194780678
1	1.0e-1	915.4558791315	16.4558791315	19085.4840745715	1086.1507412382
1	1.0e-2	909.5656710509	10.5656710509	18628.5414884534	629.2081551200
1	1.0e-3	898.8842089088	-0.1157910912	18010.6077472358	11.2744139024
1	1.0e-4	901.9322095379	2.9322095379	18193.0501290685	193.7167957352
1	1.0e-5	904.2464438259	5.2464438259	18311.7658395882	312.4325062549
1	1.0e-6	901.7170043003	2.7170043003	18162.1377940041	162.8044606707
1	1.0e-7	899.9780216660	0.9780216660	18058.8227863796	59.4894530463
1	1.0e-8	899.0631521924	0.0631521924	18003.0714959720	3.7381626387
1	1.0e-9	899.1879448100	0.1879448100	18010.6662264963	11.3328931629
1	1.0e-10	899.0660416443	0.0660416443	18003.2759414757	3.9426081424
1	1.0e-11	899.0410569131	0.0410569131	18001.7926990668	2.4593657335
1	1.0e-12	899.0165782786	0.0165782786	18000.3234598418	0.9901265084
1	1.0e-13	899.0066129166	0.0066129166	17999.7259214523	0.3925881189
1	1.0e-14	899.0024919762	0.0024919762	17999.4784832192	0.1451498859

Tabelle C.28: Fehler im Fall Aufsprung mit Polynomgrad 1

Grad	eps_v	Zeit	Fehler-Zeit	Ort	Fehler-Ort
2	1.0e2	-1.0000000000	-900.0000000000	-1.0000000000	-18000.3333333333
2	1.0e1	959.3831307334	60.3831307334	20000.2803038626	2000.9469705293
2	1.0e0	893.2094996958	-5.7905003042	18015.2818754017	15.9485420684
2	1.0e-1	894.7248410069	-4.2751589931	17906.4371617512	-92.8961715821
2	1.0e-2	893.7159812526	-5.2840187474	17789.9700172641	-209.3633160692
2	1.0e-3	898.7115760809	-0.2884239191	17999.2905820238	-0.0427513095
2	1.0e-4	899.4054789868	0.4054789868	18024.8360367334	25.5027034000
2	1.0e-5	898.8254071503	-0.1745928497	17991.6125059297	-7.7208274036
2	1.0e-6	898.9787111904	-0.0212888096	17998.4895860048	-0.8437473285
2	1.0e-7	899.0356101332	0.0356101332	18001.1757349121	1.8424015787
2	1.0e-8	898.9967942635	-0.0032057365	17999.1811561991	-0.1521771342
2	1.0e-9	899.0029646067	0.0029646067	17999.4806001702	0.1472668369
2	1.0e-10	898.9992951657	-0.0007048343	17999.2952798051	-0.0380535283
2	1.0e-11	899.0001188577	0.0001188577	17999.3360926676	0.0027593343
2	1.0e-12	899.0000946288	0.0000946288	17999.3348391751	0.0015058418
2	1.0e-13	899.0001141091	0.0001141091	17999.3358061437	0.0024728103
2	1.0e-14	899.0001050735	0.0001050735	17999.3353531871	0.0020198538

Tabelle C.29: Fehler im Fall Aufsprung mit Polynomgrad 2

Grad	eps_v	Zeit	Fehler-Zeit	Ort	Fehler-Ort
3	1.0e2	-1.0000000000	-900.0000000000	-1.0000000000	-18000.3333333333
3	1.0e1	953.7630270114	54.7630270114	20000.2803038626	2000.9469705293
3	1.0e0	889.2252810604	-9.7747189396	17835.8463865785	-163.4869467548
3	1.0e-1	895.7827049850	-3.2172950150	17954.2161832103	-45.1171501230
3	1.0e-2	899.0653895026	0.0653895026	18036.7116954761	37.3783621428
3	1.0e-3	898.6871354596	-0.3128645404	17997.9970884658	-1.3362448675
3	1.0e-4	898.9984887995	-0.0015112005	18003.1065539246	3.7732205912
3	1.0e-5	898.9688278002	-0.0311721998	17998.7461914206	-0.5871419127
3	1.0e-6	898.9915812073	-0.0084187927	17999.1453679198	-0.1879654135
3	1.0e-7	899.0000044458	0.0000044458	17999.3757044051	0.0423710718
3	1.0e-8	899.0000532762	0.0000532762	17999.3416850836	0.0083517503
3	1.0e-9	899.0000983567	0.0000983567	17999.3366456994	0.0033123661
3	1.0e-10	899.0001088621	0.0001088621	17999.3358132505	0.0024799171
3	1.0e-11	899.0001059722	0.0001059722	17999.3354517835	0.0021184502
3	1.0e-12	899.0001075772	0.0001075772	17999.3354869776	0.0021536442
3	1.0e-13	899.0001078360	0.0001078360	17999.3354924759	0.0021591426
3	1.0e-14	899.0001078757	0.0001078757	17999.3354932585	0.0021599252

Tabelle C.30: Fehler im Fall Aufsprung mit Polynomgrad 3

Grad	eps_v	Zeit	Fehler-Zeit	Ort	Fehler-Ort
4	1.0e2	-1.0000000000	-900.0000000000	-1.0000000000	-18000.3333333333
4	1.0e1	954.3530802602	55.3530802602	20000.2803038626	2000.9469705293
4	1.0e0	889.9702967047	-9.0297032953	17862.9796226532	-136.3537106801
4	1.0e-1	896.0577158826	-2.9422841174	17965.4740929084	-33.8592404250
4	1.0e-2	897.6195887949	-1.3804112051	17976.5766174171	-22.7567159163
4	1.0e-3	898.6833962137	-0.3166037863	17997.8158727204	-1.5174606129
4	1.0e-4	898.9284890843	-0.0715109157	17999.6869295752	0.3535962419
4	1.0e-5	898.9816858925	-0.0183141075	17999.3336802121	0.0003468787
4	1.0e-6	898.9955828847	-0.0044171153	17999.3298141245	-0.0035192089
4	1.0e-7	898.9991177381	-0.0008822619	17999.3343817746	0.0010484413
4	1.0e-8	898.9999079031	-0.0000920969	17999.3350484622	0.0017151288
4	1.0e-9	899.0000729710	0.0000729710	17999.3354695709	0.0021362376
4	1.0e-10	899.0001018293	0.0001018293	17999.3354894243	0.0021560910
4	1.0e-11	899.0001068643	0.0001068643	17999.3354929753	0.0021596420
4	1.0e-12	899.0001077129	0.0001077129	17999.3354932400	0.0021599067
4	1.0e-13	899.0001078527	0.0001078527	17999.3354932484	0.0021599151
4	1.0e-14	899.0001078756	0.0001078756	17999.3354932533	0.0021599200

Tabelle C.31: Fehler im Fall Aufsprung mit Polynomgrad 4

Grad	eps_v	Zeit	Fehler-Zeit	Ort	Fehler-Ort
5	1.0e2	-1.0000000000	-900.0000000000	-1.0000000000	-18000.3333333333
5	1.0e1	954.3882275736	55.3882275736	20000.2803038626	2000.9469705293
5	1.0e0	890.1690497291	-8.8309502709	17870.5243548884	-128.8089784449
5	1.0e-1	896.0102719020	-2.9897280980	17963.6951779335	-35.6381553999
5	1.0e-2	897.9393222476	-1.0606777524	17989.2916775836	-10.0416557498
5	1.0e-3	898.6828100929	-0.3171899071	17997.7889996373	-1.5443336960
5	1.0e-4	898.9163991942	-0.0836008058	17999.1280788858	-0.2052544475
5	1.0e-5	898.9806126252	-0.0193873748	17999.2867256028	-0.0466077306
5	1.0e-6	898.9955270725	-0.0044729275	17999.3273507363	-0.0059825971
5	1.0e-7	898.9991122966	-0.0008877034	17999.3341424871	0.0008091538
5	1.0e-8	898.9999144371	-0.0000855629	17999.3353324234	0.0019990900
5	1.0e-9	899.0000730377	0.0000730377	17999.3354725170	0.0021391837
5	1.0e-10	899.0001018646	0.0001018646	17999.3354909693	0.0021576360
5	1.0e-11	899.0001068650	0.0001068650	17999.3354930065	0.0021596732
5	1.0e-12	899.0001077126	0.0001077126	17999.3354932272	0.0021598939
5	1.0e-13	899.0001078527	0.0001078527	17999.3354932506	0.0021599173
5	1.0e-14	899.0001078756	0.0001078756	17999.3354932531	0.0021599197

Tabelle C.32: Fehler im Fall Aufsprung mit Polynomgrad 5

Grad	eps_v	Zeit	Fehler-Zeit	Ort	Fehler-Ort
6	1.0e2	-1.0000000000	-900.0000000000	-1.0000000000	-18000.3333333333
6	1.0e1	954.3490835148	55.3490835148	20000.2803038626	2000.9469705293
6	1.0e0	890.1488747252	-8.8511252748	17869.9717196830	-129.3616136503
6	1.0e-1	896.0086061052	-2.9913938948	17963.6254015679	-35.7079317654
6	1.0e-2	897.8837679422	-1.1162320578	17987.1168297957	-12.2165035376
6	1.0e-3	898.6827169650	-0.3172830350	17997.7849003232	-1.5484330101
6	1.0e-4	898.9143230033	-0.0856769967	17999.0359132295	-0.2974201039
6	1.0e-5	898.9805932469	-0.0194067531	17999.2859149355	-0.0474183978
6	1.0e-6	898.9955185673	-0.0044814327	17999.3269910322	-0.0063423011
6	1.0e-7	898.9991131644	-0.0008868356	17999.3341795315	0.0008461982
6	1.0e-8	898.9999141415	-0.0000858585	17999.3353200354	0.0019867021
6	1.0e-9	899.0000730453	0.0000730453	17999.3354728380	0.0021395046
6	1.0e-10	899.0001018648	0.0001018648	17999.3354909773	0.0021576440
6	1.0e-11	899.0001068650	0.0001068650	17999.3354930053	0.0021596719
6	1.0e-12	899.0001077126	0.0001077126	17999.3354932272	0.0021598938
6	1.0e-13	899.0001078527	0.0001078527	17999.3354932506	0.0021599173
6	1.0e-14	899.0001078756	0.0001078756	17999.3354932531	0.0021599197

Tabelle C.33: Fehler im Fall Aufsprung mit Polynomgrad 6

Grad	eps_v	Zeit	Fehler-Zeit	Ort	Fehler-Ort
7	1.0e2	-1.0000000000	-900.0000000000	-1.0000000000	-18000.3333333333
7	1.0e1	954.3603483830	55.3603483830	20000.2803038626	2000.9469705293
7	1.0e0	890.1393831134	-8.8606168866	17869.6518963896	-129.6814369437
7	1.0e-1	896.0101323380	-2.9898676620	17963.6782675492	-35.6550657842
7	1.0e-2	897.8875467209	-1.1124532791	17987.2835265355	-12.0498067978
7	1.0e-3	898.6827020272	-0.3172979728	17997.7842635801	-1.5490697532
7	1.0e-4	898.9139694045	-0.0860305955	17999.0207087375	-0.3126245958
7	1.0e-5	898.9805997151	-0.0194002849	17999.2861802449	-0.0471530884
7	1.0e-6	898.9955187781	-0.0044812219	17999.3269997179	-0.0063336154
7	1.0e-7	898.9991132210	-0.0008867790	17999.3341818762	0.0008485429
7	1.0e-8	898.9999141549	-0.0000858451	17999.3353205822	0.0019872488
7	1.0e-9	899.0000730454	0.0000730454	17999.3354728443	0.0021395110
7	1.0e-10	899.0001018648	0.0001018648	17999.3354909770	0.0021576437
7	1.0e-11	899.0001068650	0.0001068650	17999.3354930053	0.0021596719
7	1.0e-12	899.0001077126	0.0001077126	17999.3354932272	0.0021598938
7	1.0e-13	899.0001078527	0.0001078527	17999.3354932506	0.0021599173
7	1.0e-14	899.0001078756	0.0001078756	17999.3354932531	0.0021599197

Tabelle C.34: Fehler im Fall Aufsprung mit Polynomgrad 7

Grad	eps_v	Zeit	Fehler-Zeit	Ort	Fehler-Ort
8	1.0e2	-1.0000000000	-900.0000000000	-1.0000000000	-18000.3333333333
8	1.0e1	954.3587722394	55.3587722394	20000.2803038626	2000.9469705293
8	1.0e0	890.1399732335	-8.8600267665	17869.6641634767	-129.6691698566
8	1.0e-1	896.0099786319	-2.9900213681	17963.6735144785	-35.6598188548
8	1.0e-2	897.8901661470	-1.1098338530	17987.3670031616	-11.9663301717
8	1.0e-3	898.6826996143	-0.3173003857	17997.7841634058	-1.5491699276
8	1.0e-4	898.9139097848	-0.0860902152	17999.0182113413	-0.3151219921
8	1.0e-5	898.9805995507	-0.0194004493	17999.2861736286	-0.0471597048
8	1.0e-6	898.9955187984	-0.0044812016	17999.3270005359	-0.0063327975
8	1.0e-7	898.9991132230	-0.0008867770	17999.3341819544	0.0008486211
8	1.0e-8	898.9999141543	-0.0000858457	17999.3353205579	0.0019872245
8	1.0e-9	899.0000730454	0.0000730454	17999.3354728444	0.0021395110
8	1.0e-10	899.0001018648	0.0001018648	17999.3354909770	0.0021576437
8	1.0e-11	899.0001068650	0.0001068650	17999.3354930053	0.0021596719
8	1.0e-12	899.0001077126	0.0001077126	17999.3354932272	0.0021598938
8	1.0e-13	899.0001078527	0.0001078527	17999.3354932506	0.0021599173
8	1.0e-14	899.0001078756	0.0001078756	17999.3354932531	0.0021599197

Tabelle C.35: Fehler im Fall Aufsprung mit Polynomgrad 8

Grad	eps_v	Zeit	Fehler-Zeit	Ort	Fehler-Ort
9	1.0e2	-1.0000000000	-900.0000000000	-1.0000000000	-18000.3333333333
9	1.0e1	954.3586065106	55.3586065106	20000.2803038626	2000.9469705293
9	1.0e0	890.1404264636	-8.8595735364	17869.6782017013	-129.6551316320
9	1.0e-1	896.0099534903	-2.9900465097	17963.6726357045	-35.6606976289
9	1.0e-2	897.8883986040	-1.1116013960	17987.3086625607	-12.0246707726
9	1.0e-3	898.6826992226	-0.3173007774	17997.7841474970	-1.5491858363
9	1.0e-4	898.9138998507	-0.0861001493	17999.0178042963	-0.3155290370
9	1.0e-5	898.9805995196	-0.0194004804	17999.2861724092	-0.0471609241
9	1.0e-6	898.9955187977	-0.0044812023	17999.3270005058	-0.0063328275
9	1.0e-7	898.9991132230	-0.0008867770	17999.3341819553	0.0008486220
9	1.0e-8	898.9999141543	-0.0000858457	17999.3353205589	0.0019872256
9	1.0e-9	899.0000730454	0.0000730454	17999.3354728444	0.0021395110
9	1.0e-10	899.0001018648	0.0001018648	17999.3354909770	0.0021576437
9	1.0e-11	899.0001068650	0.0001068650	17999.3354930053	0.0021596719
9	1.0e-12	899.0001077126	0.0001077126	17999.3354932272	0.0021598938
9	1.0e-13	899.0001078527	0.0001078527	17999.3354932506	0.0021599173
9	1.0e-14	899.0001078756	0.0001078756	17999.3354932531	0.0021599197

Tabelle C.36: Fehler im Fall Aufsprung mit Polynomgrad 9

Anhang D

Thesen

1. Die in dieser Arbeit entwickelten Algorithmen sollen als schnelles Simulationswerkzeug zur Fahrzeitrechnung zum Einsatz kommen.

   Hauptzielgruppe sind die Betriebszentralen der Deutschen Bahn.

2. Ziel war es, Dispositionshilfen für die Bearbeitung eines einzelnen Zuges und eines Systems aus mehreren, sich gegenseitig beeinflussenden Zügen zu schaffen.

3. Schwerpunkt der Entwicklung der Algorithmen war neben der obligatorischen sinnvollen Genauigkeit der Rechnung eine schnelle Laufzeit der Programme, um Testreihen für Optimierungsstrategien in effektiven Rechenzeiten absolvieren zu können.

4. Es wurden sowohl die theoretischen Grundlagen der Simulation sowie der optimalen Steuerung als auch die algorithmische Umsetzung in ein zeitgemäßes C++ Programm betrachtet.

5. Auf hohe Portabilität und ISO-Konformität des Quellcodes wurde besonderer Wert gelegt, da die spätere Einsatzplattform noch nicht feststeht.

Stichwortverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] *Vorlesungsnachschriften „Variationsrechnung und optimale Steuerung“* . Prof. Tröltzsch . TU Chemnitz, Wintersemester 1998/1999.

[2] *Vorlesungsscriptum „Variationsrechnung 2“* . Prof. Maurer . TU München, Wintersemester 1987/1988.

[3] *Vorlesungsscriptum „Numerische Methoden in der Steuerungstheorie“* . Prof. Mehrmann . TU Chemnitz, Sommersemester 1995.

[4] *Vorlesungsscriptum „Numerik gewöhnlicher Differentialgleichungen“* . Prof. Mehrmann . TU Chemnitz, Wintersemester 1996/1997.

[5] *Vorlesungsscriptum „Numerik“* . Prof. Mehrmann . TU Chemnitz, Sommersemester 1999.

[6] Robert Schaback, Helmut Werner. *Numerische Mathematik*. Springer-Verlag, Berlin, Heidelberg, New Yorck Stuttgart, 1992.

[7] Ch. Großmann, J. Terno. *Numerik der Optimierung*. Teubner, Stuttgart, 1997.

[8] Leslie M. Hocking. *Optimal Control An Introduction to the Theory with Applications*. Clarendon Press, Oxford, 1991.

[9] Jörn Pachl. *Systemtechnik des Schienenverkehrs*. Teubner, Stuttgart; Leipzig, 1999.

[10] Erwin Bude. *Grundzüge Numerik - Theorie und Aufgaben*. Verlag Shaker, Aachen, 1990.

[11] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley, Bonn, 2000.

[12] Helmut Herold. *C-Kompaktreferenz*. Addison-Wesley, Bonn, 1999.

[13] Torsten Machert. *Wissenschaftliches Publizieren mit LATEX2e*. Vieweg Verlag, Braunschweig/Wiesbaden,1998.