Technische Universitaet Berlin

Bachelorarbeit

Realitätsnahe Fahrzeugsteuerung für die Eisenbahnbetriebssimulation im Eisenbahn-Betriebs- und Experimentierfeld

Friedrich Kasper Völkers

betreut von
Dr.-Ing. Christian Blome

8. August 2021

Aufgabenstellung

Im Eisenbahn-Betriebs- und Experimentierfeld (EBuEf) des Fachgebietes Bahnbetrieb und Infrastruktur der Technischen Universität Berlin können Prozesse des Bahnbetriebs unter realitätsnahen Bedingungen simuliert werden. Den Mittelpunkt der Anlagen bilden originale Stellwerke unterschiedlicher Entwicklungsstufen der Eisenbahnsicherungstechnik vom mechanischen Stellwerk bis zu aus einer Betriebszentrale gesteuerten Elektronischen Stellwerken.

Das „Ausgabemedium“ ist eine Modellbahnanlage, die in verkleinertem Maßstab die Abläufe darstellt. Das Betriebsfeld wird in der Lehre im Rahmen der Bachelor- und Masterstudiengänge am Fachgebiet sowie darüber hinaus zur Ausbildung von Fahrdienstleitern, für Schulungen und Weiterbildungen Externer sowie bei öffentlichen Veranstaltungen wie beispielsweise der Langen Nacht der Wissenschaften eingesetzt.

Neben den Stellwerken ist auch bei den Fahrzeugen ein möglichst realitätsnaher Betrieb Teil der umfassenden Eisenbahnbetriebssimulation.

Ziel dieser Arbeit ist die Entwicklung einer Steuerungssoftware, die auf dem (modellseitig nur) punktförmig überwachten Netz die Fahrzeuge kontinuierlich überwacht, um die Fahrzeuge realitätsnäher zu steuern (beispielsweise durch maßstäbliche Beschleunigung oder punktgenaues Anhalten an Bahnsteigen gemäß der aktuellen Zuglänge) und zukünftig auch andere und neue Betriebsverfahren wie Moving Block im EBuEf simulieren zu können.

Teil der kontinuierlichen Überwachung ist die exakte Positionsbestimmung der Fahrzeuge im Netz sowie die Übermittlung der aktuellen Geschwindigkeit.

Beschleunigungs- und Bremsvorgänge sowie Ausrollphasen für optional energieoptimales Fahren sind ebenso zu berücksichtigen. Zur Kalibrierung sind die schon vorhandenen Ortungsmöglichkeiten (Belegung von Gleisabschnitten) zu verwenden.

Weitere zu berücksichtigende Eingangsgrößen aus der vorhandenen Softwarelandschaft im EBuEf sind die Netztopologie (z.B. Streckenlängen, Signalstandorte), die Fahrzeugdaten, die aktuelle Zugbildung sowie die Prüfung (vorhandene API), ob ein Zug an einer Station anhalten muss und ob er abfahren darf. Damit sind in der Simulation Fahrplantreue, Verspätungen sowie Personalausfälle darstellbar.

Die Erkenntnisse sind in einem umfassenden Bericht und einer zusammenfassenden Textdatei darzustellen. Darüber hinaus sind die Ergebnisse der Arbeit ggf. im Rahmen einer Vortragsveranstaltung des Fachgebiets zu präsentieren.

Der Bericht soll in gedruckter Form als gebundenes Dokument sowie in elektronischer Form als ungeschütztes PDF-Dokument eingereicht werden. Methodik und Vorgehen bei der Arbeit sind explizit zu beschreiben und auf eine entsprechende Zitierweise ist zu achten. Alle genutzten bzw. verarbeiteten zugrundeliegenden Rohdaten sowie nicht-veröffentlichte Quellen müssen der Arbeit (ggf. in elektronischer Form) beiliegen.

In dem Bericht ist hinter dem Deckblatt der originale Wortlaut der Aufgabenstellung der Arbeit einzuordnen. Weiterhin muss der Bericht eine einseitige Zusammenfassung der Arbeit enthalten. Diese Zusammenfassung der Arbeit ist zusätzlich noch einmal als eigene, unformatierte Textdatei einzureichen.

Für die Bearbeitung der Aufgabenstellung sind die Hinweise zu beachten, die auf der Webseite mit der Adresse www.railways.tu-berlin.de/?id=66923 gegeben werden.

Der Fortgang der Abarbeitung ist in engem Kontakt mit dem Betreuer regelmäßig abzustimmen. Hierzu zählen insbesondere mindestens alle vier Wochen kurze Statusberichte in

mündlicher oder schriftlicher Form.

Zusammenfassung

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ICE Inter City Express

EBuEf Eisenbahn-Betriebs- und Experimentierfeld

RE Regional Express

PZ Personenzug

GZ Güterzug

1 Inhalt

- Programmablauf
- Grundlagen zur linienförmigen Zugbeeinflussung, moving-block-Verfahren etc.
- Aufbau der Tabelle
- Formeln, Herkunft, Ableitung, Vereinfachung und Annahmen etc.
- Beschreibung der Methoden
- Was sind Ziele, Grundlagen oder Rahmenbedingung, auf die immer geachtet wird (Skalierbarkeit, Erweiterbar, Wenig „traffic“ in der Datanbank, Einheitlichkeit etc.)
- Was wird benötigt? (SQL, php etc.)
- Struktogramm (`http://rhinodidactics.de/Artikel/latex3.html`)
- Umgang mit dem Auf- und Abrunden bei Kommazahlen (Zeitangaben z.B.)
- möglichst Energiesparend, also so langsam, dass der Zug gerade noch so pünktlich ankommt

2 Grundlagen

2.1 Zitate

At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. L consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet.[1], [2] Lorem ipsum dolor sit amem nonumore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumgnoluptua. At vero eos lor sit amet. Loreonumy eirmod tempor invidunt ut labore et dolore magna aliquyam clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet.[3], [4]

2.2 Abkürzungen

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, Regional Express (RE) sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et Inter City Express (ICE) dolore magna aliquyam erat, sed diam Personenzug (PZ) voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet PZ.

[1] Maschek (2013)
[2] Wende (2013)
[3] Maschek (2013)
[4] Wende (2013)

2.3 Zitat

„Hongkong must build a [...] rapid transit system, or a more expensive roads system, in the next 16 years – or face potentially devastating effects on its economy.“

2.4 Bild

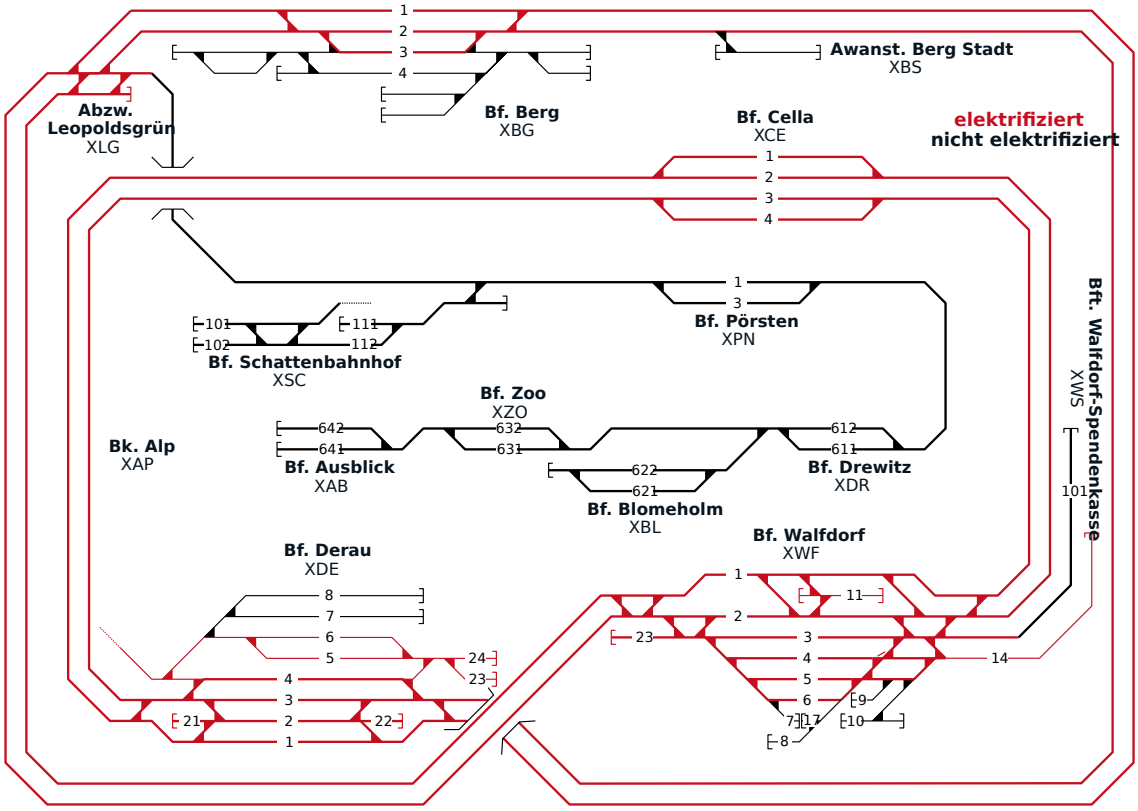

Abbildung 1: Schienennetz

2.5 Tabelle

ICE 1	ICE 2	ICE 3
200 km/h	250 km/h	300 km/h

Tabelle 1: Sehr sehr schöne Tabelle

Bezeichnung	Funktion
$keyPoint (Array)	Beschreibt eine Beschleunigung bzw. Verzögerung (*position_0*, *position_1*, *time_0*, *time_1*, *speed_0*, *speed_1*)
$next_section (Array)	IDs aller Abschnitte
$next_lenghts (Array)	Längen aller Abschnitte
$next_v_max (Array)	Höchstgeschwindigkeit aller Abschnitte
$indexCurrentSection (Integer)	Index des aktuellen Abschnitts
$indexTargetSection (Integer)	Index des Ziel-Abschnitts
$cumulativeSectionLengthStart (Array)	Absolute Startposition aller Abschnitte
$cumulativeSectionLengthEnd (Array)	Absolute Endposition aller Abschnitte
$trainPositionChange (Array)	Alle absoluten Positionen des Fahrtverlaufs
$trainSpeedChange (Array)	Alle Geschwindigkeiten des Fahrtverlaufs

Tabelle 2: Beschreibung der verwendeten Variablen für die Fahrtverlaufsberechnung

3 Berechnung des Fahrtverlaufs

Der Fahrtverlauf eines Fahrzeuges wird bei der Berechnung in zwei verschiedenen Arten gespeichert. Einmal in so genannten *$keyPoints*, welche in einem Array die Start- und Zielgeschwindigkeit (*time_0* und *time_0*), die Start- und Endposition (*position_0* und *position_1*) und die Start- und Endzeit (*time_0* und *time_1*) einer Beschleunigung bzw. Verzögerung abspeichern. Für die Überprüfung, ob ein Fahrzeug die zulässige Höchstgeschwindigkeit in einem Infrastrukturabschnitt überschreitet, für die spätere Übermittlung der Echtzeitdaten an das Fahrzeug und die exakte Positionsbestimmung, werden mittels der *$keyPoints* für jede Geschwindigkeitsänderungen (und bei konstanter Geschwindigkeit in 1 Meter Abständen) die aktuelle relative Position innerhalb eines Infrastrukturabschnitts, der Infrastrukturabschnitt, die aktuelle Zeit und die aktuelle Geschwindigkeit in einem Array gespeichert. Der Fahrtverlauf wird mit der Funktion *updateNextSpeed*() berechnet, welche als Parameter unter anderem die Zugdaten aus dem *$allUsedTrains*-Array, Start- und Endzeit der Fahrt (*$startTime* und *$endTime*), den Ziel-Infrastrukturabschnitt (*$targetSection*) und die relative Position in dem Ziel-Infrastrukturabschnitt (*$targetPosition*) übergeben bekommt.

Die für die Berechnung benötigten Daten werden in den in Tabelle 2 beschriebenen Variablen gespeichert.

In dem folgenden Abschnitt werden die einzelnen Schritte beschrieben, die durchlaufen werden um den optimalen Fahrtverlauf zu berechnen. In der Darstellung 2 wird der Ablauf grob schematisch dargestellt.

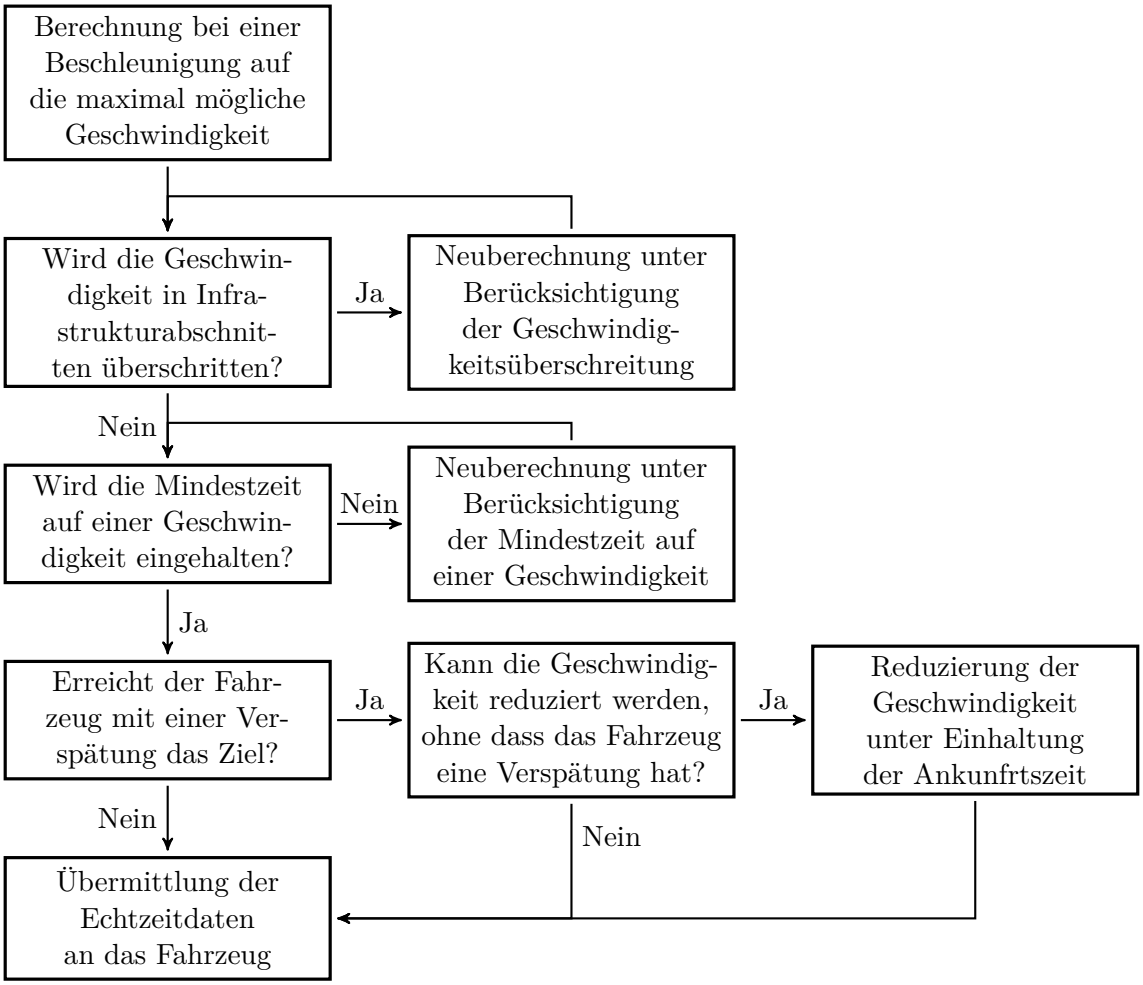

Abbildung 2: Ablaufplan der Fahrtverlaufsberechnung

Infrastrukturabschnitts-ID	Länge	zulässige Höchstgeschwindigkeit
1000	300 m	120 km/h
1001	400 m	120 km/h
1002	300 m	120 km/h
1003	400 m	90 km/h
1004	300 m	60 km/h
1005	200 m	60 km/h
1006	400 m	90 km/h
1007	500 m	120 km/h
1008	300 m	120 km/h
1009	400 m	100 km/h
1010	300 m	60 km/h
1011	300 m	40 km/h

Tabelle 3: Exemplarische Infrastrukturabschnitte

3.1 Ermittlung der Start- und Endposition der einzelnen Infrastrukturabschnitte unter Berücksichtigung der Zuglänge

Für die Berechnung eines exemplarischen Fahrtverlaufs wurden die in Tabelle 3 definierten Infrastrukturabschnitte benutzt. Diese Abschnitte wurden so gewählt, sodass alle Funktionen und die Allgemeingültigkeit des Algorithmus gezeigt werden können und treten so im EBuEf nicht auf.

Als exemplarisch gewählte Zugdaten wurden die in Tabelle 4 definierten Daten verwendet.

Die zuvor ermittelten nächsten Infrastrukturabschnitte inklusive derer Längen und zulässigen Höchstgeschwindigkeit müssen für die Berechnung des Fahrtverlaufs angepasst werden, da ein Fahrzeug erst beschleunigen darf, wenn das komplette Fahrzeug in den Infrastrukturabschnitt eingefahren ist. In Darstellung 3 sind die Infrastrukturabschnitte dargestellt, so wie sie von dem Fahrzeug ermittelt wurden. Dabei werden alle Abschnitte, die das Fahrzeug schon durchfahren hat oder hinter dem Zielabschnitt liegen nicht dargestellt. Zudem wird in dem aktuellen Abshcnitt die relative Position von der Länge abgezogen und in der Zielabschnitt wird nur bis zur relativen Zielposition abgebildet. Dementsprechend ist der erste Abschnitt in der Darstellung 3 der Abschnitt mit der ID 1001. Dieser hat aufgrund der aktuellen relativen Position des Fahrzeugs eine Länge von 290 m. Und der letzte Abschnitt ist der Abschnitt mit der ID 1010 und einer Länge von ebenfalls 290 m.

Bei der Berücksichtigung der Fahrzeuglänge wird durch alle Infrastrukturabschnitt iteriert und die Zuglänge auf die Länge das Abschnitts addiert. Von dieser neu ermittelten Endposition des Abschnitts wird überprüft, ob zwischen der vorherigen Endposition und der neu ermittelten Endposition ein Infrastrukturabschnitt liegt, dessen zulässige Höchstgeschwindigkeit geringer ist, als die des ursprünglichen Abschnitts. Wenn dieser Fall eintritt, wird der Abschnitt nur so weit verlängert, sodass keine Höchstgeschwindigkeit der folgenden Abschnitte überschritten wird. Von der neu ermittelten Endposition wird überprüft, in welchem Abschnitt diese liegt und mit dem Abschnitt wird dann weiter gerechnet. Sobald der

relative Startposition	10 m
relative Zielposition	290 m
aktueller Infrastrukturabschnitt	1001
Ziel-Infrastrukturabschnitt	1010
Startgeschwindigkeit	0 km/h
Zielgeschwindigkeit	0 km/h
Zuglänge	50 m
Bremsverzögerung	0,8 m/s^2
Fahrplan vorhanden	ja
Zeit bis zur nächsten Betriebsstelle	210 s

Tabelle 4: Exemplarische Zugdaten

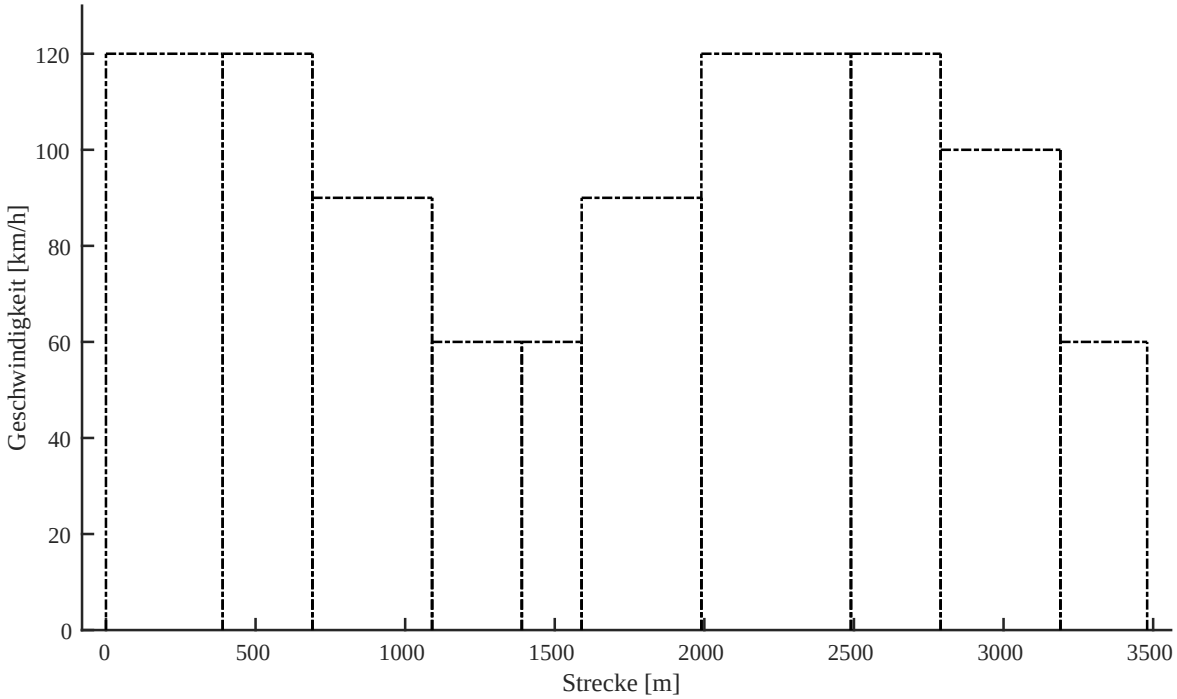

Abbildung 3: Darstellung der Infrastrukturabschnitte und die zugehörige Höchstgeschwindigkeit

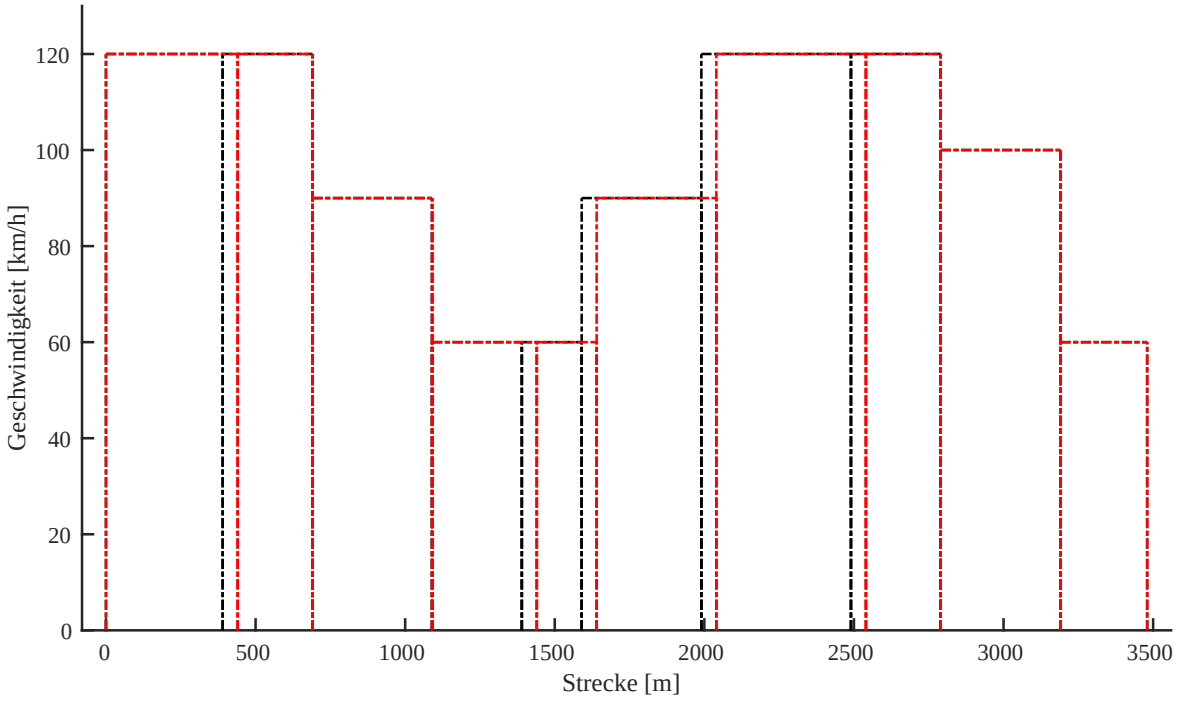

Abbildung 4: Darstellung Infrastrukturabschnitte und die zugehörige Höchstgeschwindigkeit unter Berücksichtigung der Fahrzeuglänge

Ziel-Abschnitt erreicht wurde, wird die Schleife abgebrochen. Die neu ermittelten Abschnitte werden in den Arrays *$next_lengths_mod* und *$next_v_max_mod* abgespeichert. Durch diese Algorithmus kann es dazu kommen, dass sich die Anzahl der Abschnitte verändert hat. Dementsprechend können die Abschnitte nicht mehr eindeutig mit der Infrastruktur-ID bezeichnet werden. Mittels *$next_lengths_mod* und *$next_v_max_mod* werden mit der Funktion *createCumulativeSections*() für jeden Abschnitt die absolute Start- und Endposition in den Arrays *$cumulativeSectionLengthStartMod* und *$cumulativeSectionLengthEndMod* gespeichert. Diese Umwandlung ist essentiell für die Überprüfung, in welchem Abschnitt ein Fahrzeug sich aktuell befindet. Die neu berechneten Abschnitte werden sind in der Darstellung 4 in rot abgebildet und beschreiben die maximale Geschwindigkeit, die ein Fahrzeug fahren darf an der jeweiligen Position.

3.2 Berechnung bei einer Beschleunigung auf die maximal mögliche Geschwindigkeit

Im ersten Schritt für die Distanz zwischen der aktuellen Position und der Ziel-Position mittels *$cumulativeSectionLengthStart*, *$cumulativeSectionLengthEnd*, *$indexCurrentSection* und *$indexTargetSection* berechnet. Für diese Distanz und die Startgeschwindigkeit wird mit Hilfe der Funktion *getVMaxBetweenTwoPoints*() (Code-Beispiel 1) die maximale Geschwindigkeit ermittelt, die das Fahrzeug aufnehmen kann, um noch bis zum Ziel rechtzeitig bremsen zu können. Dabei wird in 10 km/h-Schritten iteriert und der maximale Wert zurückgegeben. Innerhalb der Funktion wir die Funktion *getBrakeDistance*() (Code-Beispiel 2) aufgerufen, welche die benötigte Distanz für eine Beschleunigung bzw. Verzögerung berechnet.

Durch die gegebene Startgeschwindigkeit und die höchstmögliche Geschwindigkeit wird

```php
function getVMaxBetweenTwoPoints(float $distance, int $v_0, int $v_1) {
  global $verzoegerung;
  global $globalFloatingPointNumbersRoundingError;

  $v_max = array();
  for ($i = 0; $i <= 120; $i = $i + 10) {
    if ((getBrakeDistance($v_0, $i, $verzoegerung) + getBrakeDistance($i, $v_1,
        ↪ $verzoegerung)) < ($distance + $globalFloatingPointNumbersRoundingError)) {
      array_push($v_max, $i);
    }
  }
  if (sizeof($v_max) == 0) {
    if ($v_0 == 0 && $v_1 == 0 && $distance > 0) {
      echo "Der zug müsste langsamer als 10 km/h fahren, um das Ziel zu erreichen.";
    } else {
      // TODO: Notbremsung
    }
  } else {
    if ($v_0 == $v_1 && max($v_max) < $v_0) {
      $v_max = array($v_0);
    }
  }
  return max($v_max);
}
```

Code-Beispiel 1: *getVMaxBetweenTwoPoints*()

```php
function getBrakeDistance (float $v_0, float $v_1, float $verzoegerung) {
  if ($v_0 > $v_1) {
    return $bremsweg = 0.5 * ((pow($v_0/3.6,2)-pow($v_1/3.6, 2))/($verzoegerung));
  } if ($v_0 < $v_1) {
    return $bremsweg = -0.5 * ((pow($v_0/3.6,2)-pow($v_1/3.6, 2))/($verzoegerung));
  } else {
    return 0;
  }
}
```

Code-Beispiel 2: *getBrakeDistance*()

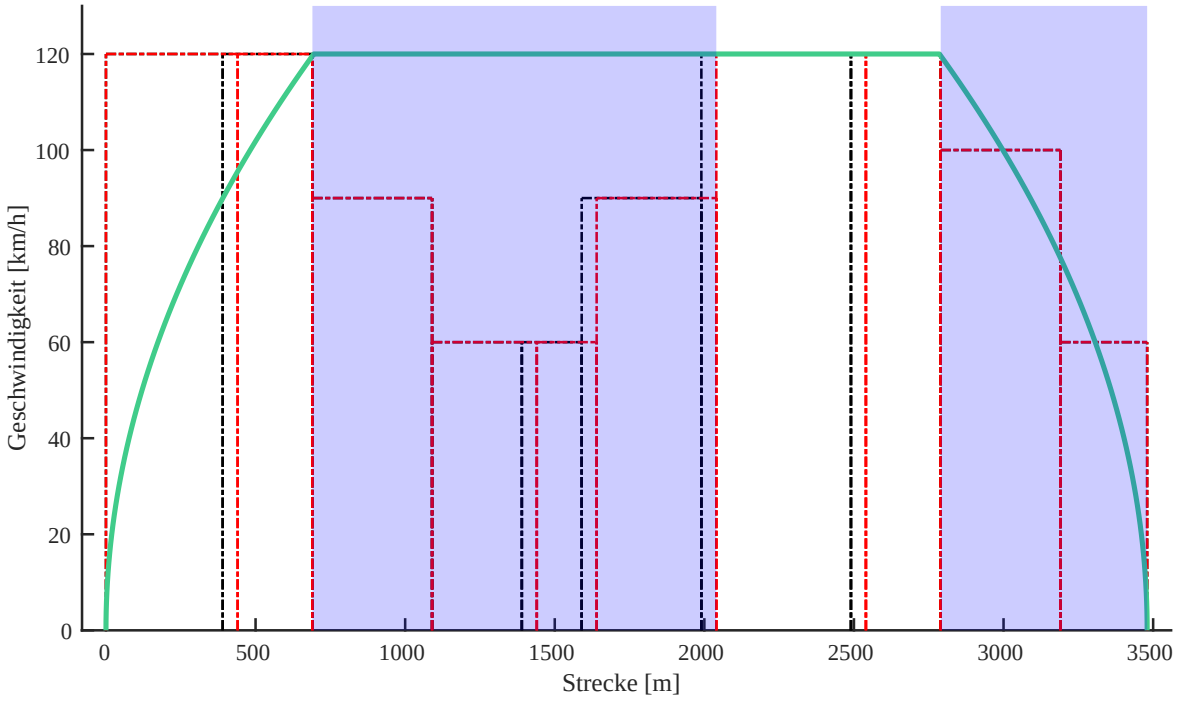

Abbildung 5: Fahrtverlaufberechnung (1. Iteration)

ein erster Fahrtverlauf berechnet. Dabei werden zwei *$keyPoints* erzeugt. Mithilfe der Funktion *createTrainChanges*() wird aus diesen beiden *$keyPoints* für jede Geschwindigkeitsveränderung die aktuelle absolute Position und Geschwindigkeit ermittelt. An den Positionen, an den das Fahrzeug eine konstante Geschwindigkeit hat, wird in 1 Meter Abständen die absolute Position und die Geschwindigkeit gespeichert. Die ermittelten Daten werden in den Arrays *$trainPositionChange* und *$trainSpeedChange* gespeichert. In der Darstellung 5 ist das Ergebnis der 1. Iteration abgebildet.

3.3 Überprüfung des Fahrtverlaufs nach Geschwindigkeitsüberschreitungen

Für die Überprüfung, ob bei einem Fahrtverlauf in manchen Infrastrukturabschnitten die zulässige Höchstgeschwindigkeit überschritten wird, wird nach jeder Berechnung die Funktion *checkIfTrainIsToFastInCertainSections*() (Code-Beispiel 3) aufgerufen. In dieser Funktion wird über alle absoluten Positionen (*$trainPositionChange*) iteriert, überprüft in welchem Abschnitt sich diese Position befindet und überprüft, ob die zugehörige Geschwindigkeit aus dem *$trainSpeedChange*-Array die zulässige Höchstgeschwindigkeit überschreitet. Sobald in einem Abschnitt eine Geschwindigkeitsüberschreitung vorliegt, wird der zugehörige Index des Abschnitts in dem *$faildSections*-Array gespeichert. Diese Abschnitte sind in der Darstellung 5 Lila hinterlegt. Als Rückgabewert der Funktion wird wird ein Array wiedergegeben, welches abspeichert, ob es zu einer Geschwindigkeitsüberschreitung gekommen ist (*"failed"*) und wenn das der Fall ist auch die Indexe der Abschnitte (*"failed_sections"*).

```php
function checkIfTrainIsToFastInCertainSections() {
  global $trainPositionChange;
  global $trainSpeedChange;
  global $cumulativeSectionLengthStartMod;
  global $next_v_max_mod;
  global $indexTargetSectionMod;

  $faildSections = array();

  foreach ($trainPositionChange as $trainPositionChangeKey =>
      ↪ $trainPositionChangeValue) {
    foreach ($cumulativeSectionLengthStartMod as $cumulativeSectionLengthStartKey =>
        ↪ $cumulativeSectionLengthStartValue) {
      if ($trainPositionChangeValue < $cumulativeSectionLengthStartValue) {
        if ($trainSpeedChange[$trainPositionChangeKey] >
            ↪ $next_v_max_mod[$cumulativeSectionLengthStartKey - 1]) {
          array_push($faildSections, ($cumulativeSectionLengthStartKey -1));
        }
        break;
      } else if ($cumulativeSectionLengthStartKey == $indexTargetSectionMod) {
        if ($trainPositionChangeValue > $cumulativeSectionLengthStartValue) {
          if ($trainSpeedChange[$trainPositionChangeKey] >
              ↪ $next_v_max_mod[$cumulativeSectionLengthStartKey]) {
            array_push($faildSections, $cumulativeSectionLengthStartKey);
          }
          break;
        }
      }
    }
  }

  if (sizeof($faildSections) == 0) {
    return array("failed" => false);
  } else {
    return array("failed" => true, "failed_sections" => array_unique($faildSections));
  }
}
```

Code-Beispiel 3: *checkIfTrainIsToFastInCertainSections*()

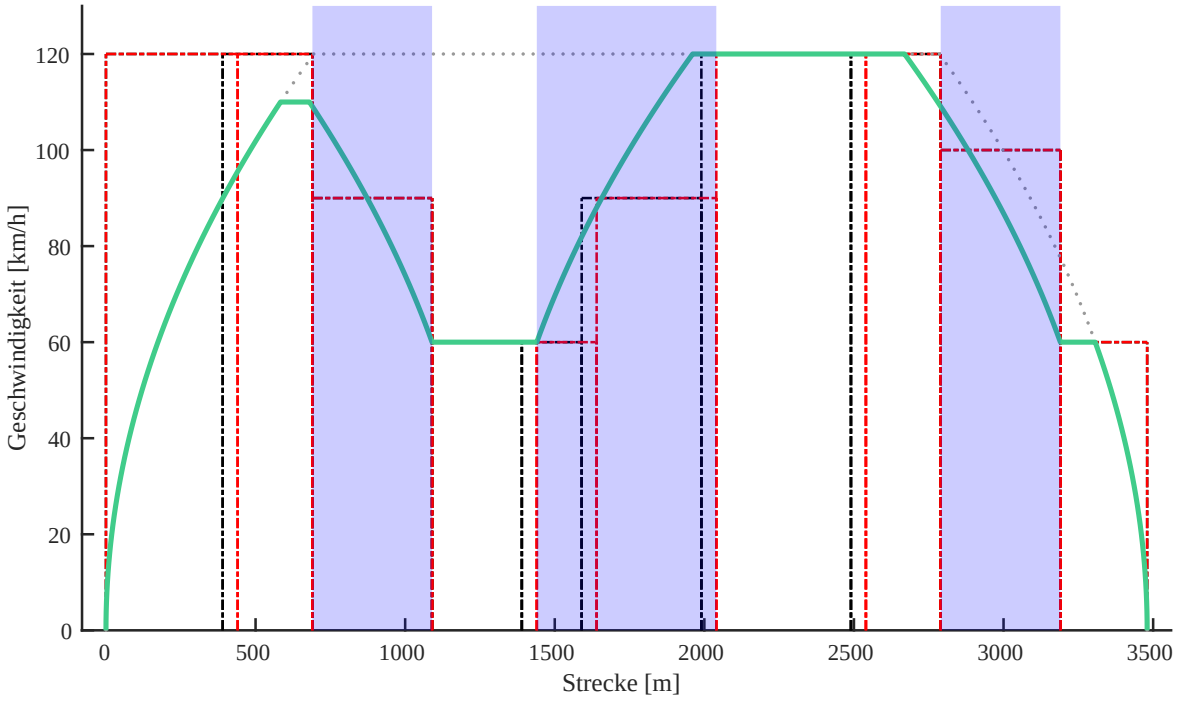

Abbildung 6: Fahrtverlaufberechnung (2. Iteration)

3.4 Neuberechnung unter Berücksichtigung der Geschwindigkeitsüberschreitung

In dem Fall, dass es zu einer Geschwindigkeitsüberschreitung gekommen ist, wird der Fahrtverlauf neu berechnet. Als Grundlage dafür diesen die *"failed_sections"* aus der *checkIfTrain IsToFastInCertainSections*() Funktion (Code-Beispiel 3). Die Funktion *recalculateKeyPoints*() vergleicht immer zwei benachbarte *$keyPoints* und berechnet in dem Fall einer Geschwindigkeitsüberschreitung mit der Funktion *checkBetweenTwoKeyPoints*() diese neu. In dem Fall, dass zwischen zwei benachbarten *$keyPoints* die zulässige Höchstgeschwindigkeit überschritten wird, wird die absolute Start- und End-Position dieser Geschwindigkeitsüberschreitung gespeichert. Im folgenden Schritt wird wie in dem Abschnitt 3.2 zwischen den Start-Werten des ersten *$keyPoints* und der ersten Geschwindigkeitsüberschreitung die maximale Geschwindigkeit berechnet und zwei neue *$keyPoints* erzeugt. Das gleiche passiert zwischen der Position der letzten Geschwindigkeitsüberschreitung und den End-Werten des zweiten *$keyPoints*. Dadurch wird sichergestellt, dass es immer eine gerade anzahl an *$keyPoints* gibt und somit in jedem Iterationsschritt zwei benachbarte *$keyPoints* verglichen werden können. Nachdem alle *$keyPoint*-Paare überprüft werden, werden mit Hilfe der *createTrainChanges*() Funktion die Arrays *$trainPositionChange* und *$trainSpeedChange* erzeugt. Dieser neu berechnete Fahrtverlauf wird dann wieder der Funktion *checkIfTrain IsToFastInCertainSections*() Funktion (Code-Beispiel 3) übergeben. Dieser Prozess wird solange durchlaufen, bis es zu keiner Geschwindigkeitsüberschreitung mehr kommt. In den folgenden Abbildungen (Darstellung 6, 7 und 8) werden die Ergebnisse der einzelnen Iterationsschritte visuell abgebildet, wobei die grau gepunkteten Linien die Ergebnisse der vorherigen Iterationsschritte darstellen.

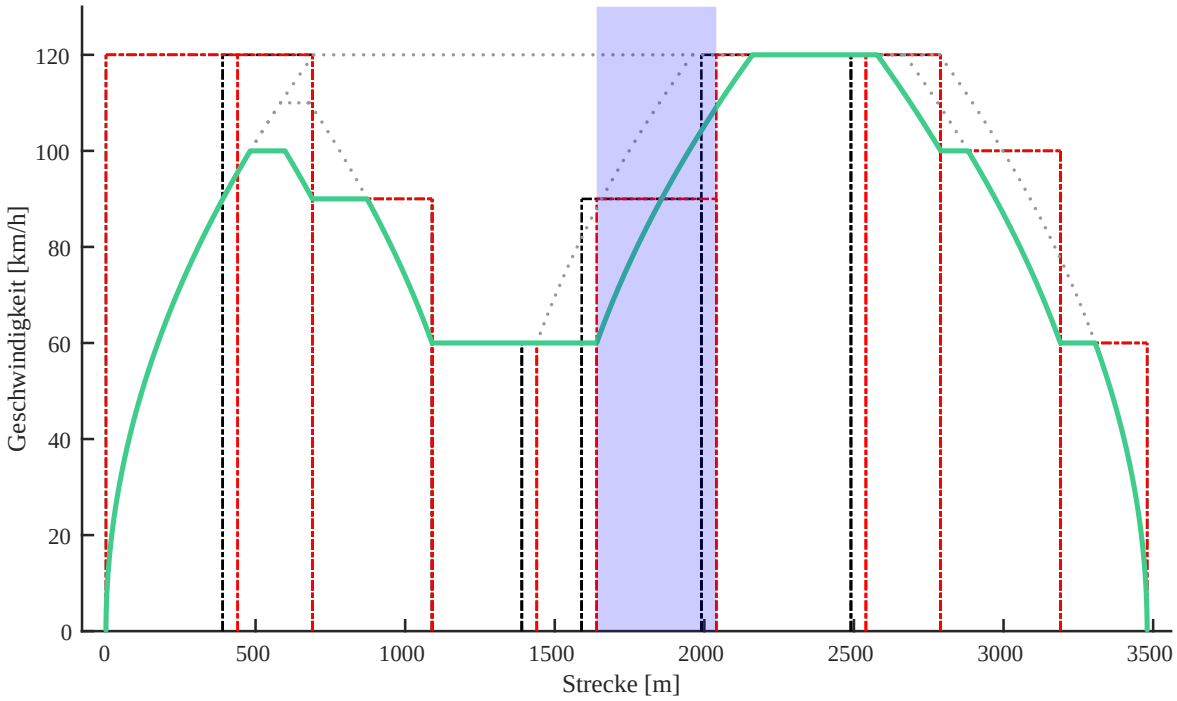

Abbildung 7: Fahrtverlaufberechnung (3. Iteration)

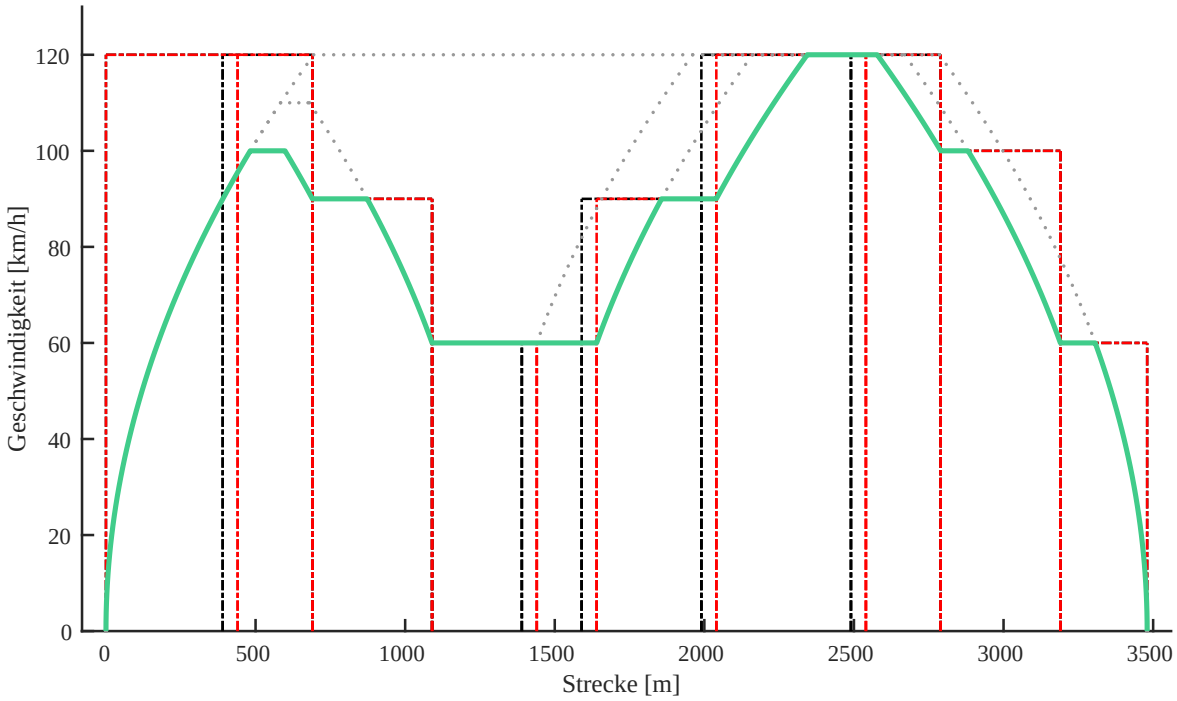

Abbildung 8: Fahrtverlaufberechnung (4. Iteration)

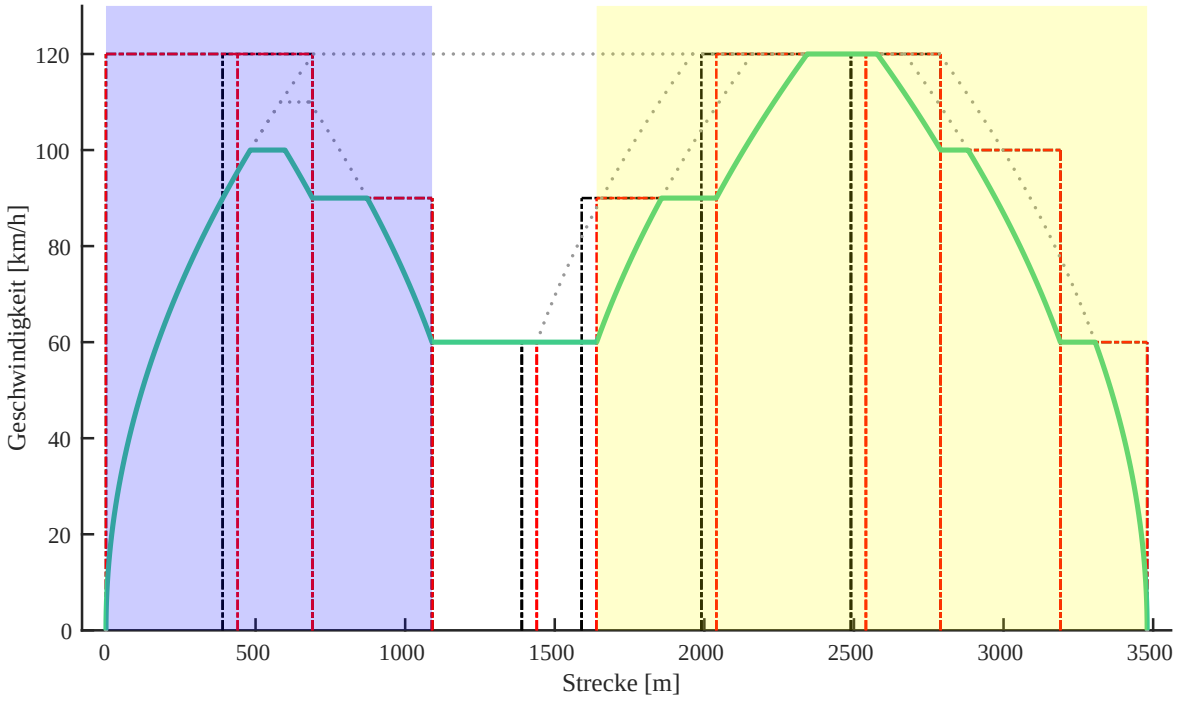

Abbildung 9: Einteilung des Fahrtverlaufs in *$subsections*

3.5 Einhaltung der Mindestzeit auf einer Geschwindigkeit

1. Ideal: möglichst späte v reduzieren

Für eine möglichst realitätsnahe Simulation kann über die Variable *$globalTimeOnOneSpeed* in der Datei *globalVariables.php* eine Mindestzeit festgelegt werden, die ein Fahrzeug auf einer Geschwindigkeit mindestens einhalten muss. Ebenfalls kann über die Variablen *$useMinTimeOnSpeed* und *$errorMinTimeOnSpeed* festgelegt werden, ob die Funktion aktiviert sein soll und ob es in dem Fall, dass diese Zeit nicht eingehalten werden kann, zu einer Fehlermeldung kommen soll. Im Falle einer Fehlermeldung würde das Fahrzeug nicht losfahren bzw. eine Gefahrenbremsung einleiten, falls das Fahrzeug aktuell eine Geschwindigkeit $v > 0$ hat.

Wenn auf einem Abschnitt die Mindestzeit nicht eingehalten werden kann, kann eine Beschleunigung später eingeleitet werden, eine Verzögerung vorzeitiger eingeleitet werden oder auf eine kleinere Geschwindigkeit beschleunigt werden. In dem folgenden Algorithmus werden die ...

Dadurch, dass sich eine Verschiebung einer Beschleunigung bzw. Verzögerung auf die nächsten Abschnitte auswirken kann, wird der Fahrtverlauf in *$subsections* unterteilt. Eine *$subsection* beschreibt dabei den Bereich des Fahrtverlaufs, in dem das Fahrzeug zum ersten Mal beschleunigt und zum letzten Mal abbremst. In der Darstellung 9 wurde der exemplarische Fahrtverlauf somit in zwei *$subsection* unterteilt, welche Lila bzw. Gelb hinterlegt sind. Diese Einteilung wird vorgenommen, da sich die Verschiebung einer Beschleunigung bzw. Verzögerung auf die folgenden bzw. vorherigen Abschnitte auswirkt. Durch diese Einteilung kann verhindert werden, dass es dadurch zu Konflikten kommt. Falls die Beschleunigungen bzw. Verzögerungen soweit nach hinten bzw. nach vorne verschoben werden müssen, kann

Index	Funktion
max_index	ndex des *$keyPoints* mit der Beschleunigung auf die maximale Geschwindigkeit in der *$subsection*
indexes	Indexe aller beinhalteten *$keyPoints*
is_prev_section	Berücksichtigung des Abschnitts vor der *$subsection*
is_next_section	Berücksichtigung des Abschnitts nach der *$subsection*
failed	Unterschreitung der Mindestzeit auf der *$subsection*
brakes_only	.....

Tabelle 5: Aufbau des *$subsection*-Arrays

die maximale Geschwindigkeit auf dieser *$subsection* reduziert werden und die zur Verfügung stehende Strecke vergrößert werden. Wie in Darstellung 9 zu erkennen wird hierbei im ersten Schritt der Abschnitt zwischen zwei *$subsections* ausgelassen. Nach der Ermittlung der *subsections* wird überprüft, ob auf den Abschnitten zwischen den *$subsections* die Mindestzeit eingehalten wird. Wenn das nicht der Fall ist, wird der Abschnitt automatisch dem in Fahrtrichtung hinteren *$subsection* zugeordnet. Dadurch wird sichergestellt, dass das Fahrzeug, wenn es an einer Stelle des Fahrtbverlaufs die Geschwindigkeit reduziert, dies möglichst spät tut.

Nachdem die *$subsections* mittels der Funktion *createSubsections*() erstellt wurden und mit der Funktion *array_reverse*() in umgekerte Reihenfolge in dem Array *$subsection_list* gesammelt wurden, wird für jede *$subsection* überprüft, ob die Beschleunigungen bzw. Verzögerungen verschoben werden können. Dabei wird über alle konstanten Geschwindigkeiten iteriert, überprüft, ob die Mindestzeit eingehalten wird und wenn das nicht der Fall ist, wird überprüft, ob eine Verschiebung möglich ist. Sollte bei einer Verschiebung die *position_*1 des *$keyPoints* hinter *position_*0 des zweiten *$keyPoints* liegen (bei einer Beschleunigung), wird der zweite *$keyPoint* gelöscht. Gleiches geschieht bei der Verzögerung in umgekehrter Reihenfolge. Nach der Verschiebung wird überprüft, ob auf allen konstanten Geschwindigkeit die Mindestzeit eingehalten wird. Wenn das der Fall ist, wird die nächste *$subsection* überprüft. In dem Fall, dass durch die Verschiebung die Mindestueit nicht eingehalten werden kann, wird die maximale Geschcindigkeit auf dieser *$subsection* um $10km/h$ reduziert, die *$subsections* neu berechnet und erneut über alle *$subsection* iteriert. Die Neuberechnung ist notwendig, da durch die Reduzierung der Geschwindigkeit die *$subsections* anders aufgeteilt sein können.

Wenn alle *$subsections* die Mindestzeit einhalten, wird der Algorithmus beendet. In der Darstellung 10 ist der Fahrtverlauf unter Einhaltung der Mindestzeit auf einer Geschwindigkeit abgebildet.

1. $v_0 notequal 0km/h$
2. Überprüfung, ob es überhaupt möglich ist.

Für den Fall, dass das Fahrzeug auf einer Geschwindigkeit die Mindestzeit nicht einhält und als nächstes beschleunigen würde, kann die Beschleunigung später eingeleitet werden.

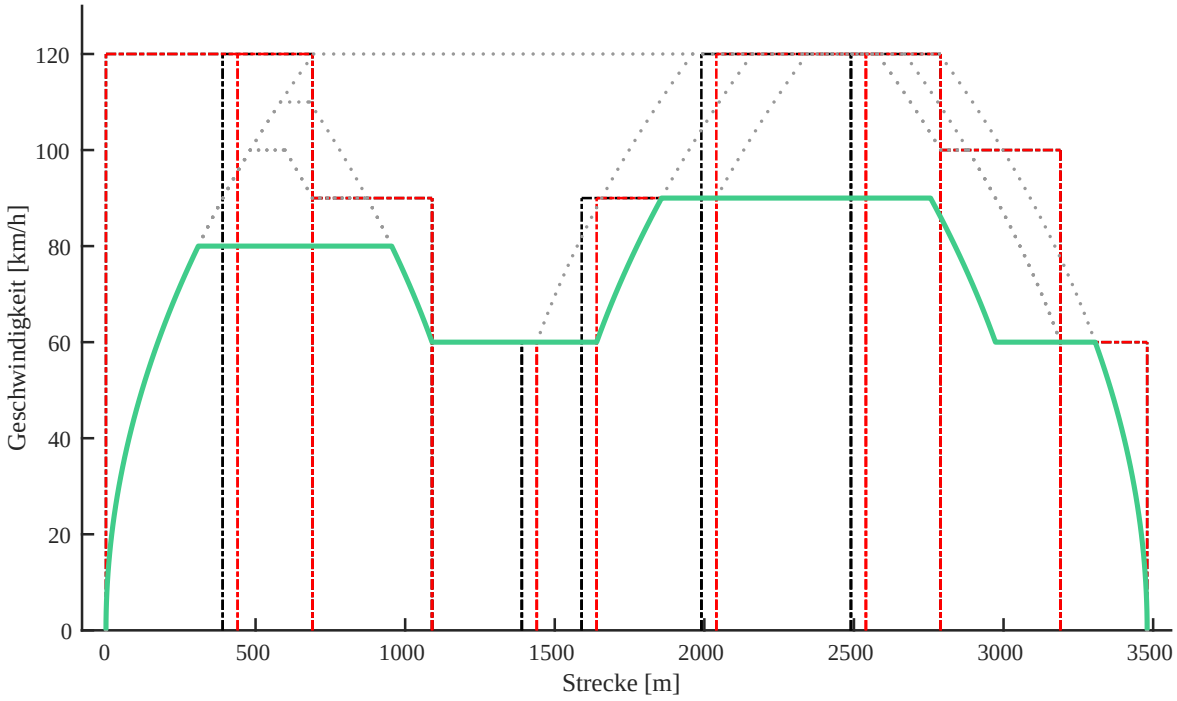

Abbildung 10: Fahrtverlauf unter Einhaltung der Mindestzeit

3.6 Berücksichtigung der Ankunfrtszeit bei der Berechnung des Fahrtverlaufs

Der berechnete Fahrtverlauf in Kapitel 3.2, 3.3, 3.4 und 3.5 ermittelt die frühstmögliche Ankunftszeit am Ziel. In dem Fall, dass der Zug dadurch mit einer Verspätung am Ziel ankommt wird der Fahrtverlauf an das Fahrzeug übergeben. Falls der Zug allerdings mit dem Fahrtverlauf zu früh am Ziel ankommen würde, wird überprüft, ob es möglich ist die Geschwindigkeit zu reduzieren, sodass der Zug energieeffizienter fahren kann und ohne Verspätung am Ziel ankommt.

Ergebnis ist in 11 abgebildet.

NACHSCHAUEN, WAS DER ALGORITHMUS GENAU MACHT!

3.7 Berücksichtigung der exakten Ankunfrtszeit bei der Berechnung des Fahrtverlaufs

Die in Kapitel 3.5 errechnete Ankunftszeit, beschreibt die spätmöglichste Ankunftszueit am Ziel, ohne dass das Fahrzeug mit einer Verspätung am Ziel ankommt, wenn bei einer Beschleunigung auf eine geringere Zielgeschwindigkeit beschleunigt wird. Dadurch wird das Fahrzeug im Normalfall noch nicht exakt pünktlich das Ziel erreichen. Über die Variable *$useSpeedFineTuning* kann festgelegt werden, ob das Fahrzeug eine exakte Ankunftsazeit versuchen soll zu erreichen.

Wenn diese Funktion aktiviert ist, wird versucht die vorletzten Verzögerung vorzeitiger einzuleiten. Für den Fall, dass es auf dem gesamten Fahrtverlauf nur eine Verzögerung gibt, wird überprüft, ob das Fahrzeug anstatt der einen Verzögerung erst auf $10km/h$ abbremsen

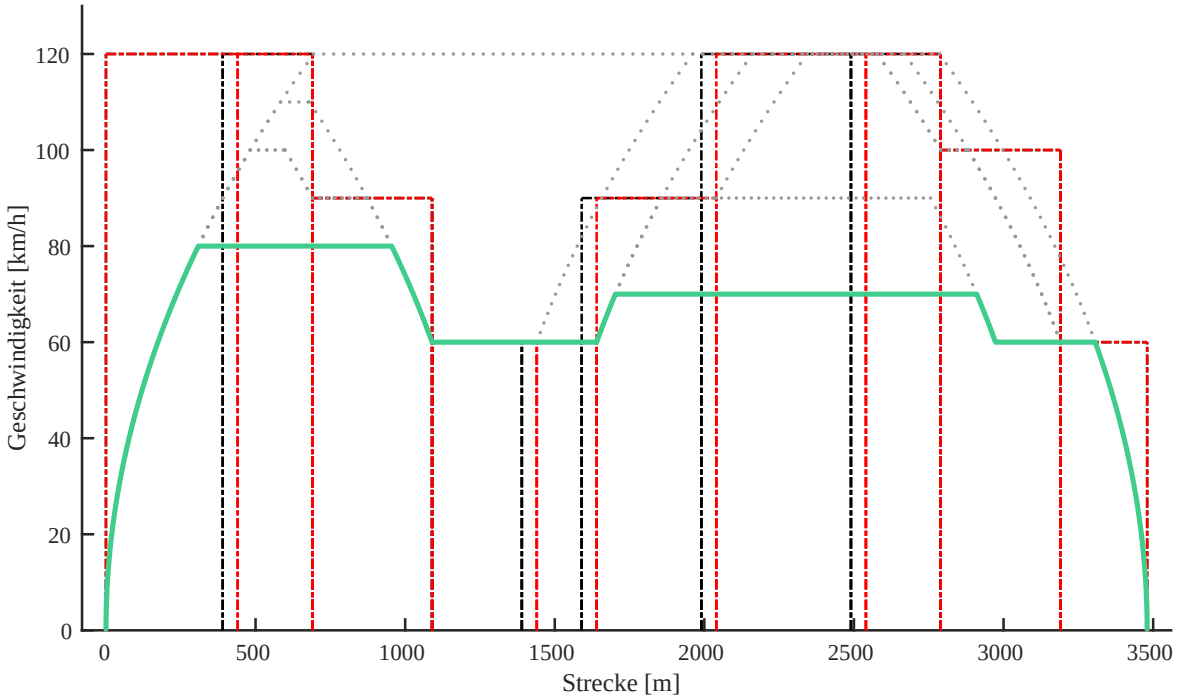

Abbildung 11: Fahrtverlauf mit reduzierter Geschwindigkeit unter Einhaltung der Ankunftszeit

kann und und kurz vorm Ziel von $10km/h$ auf $0km/h$ abbremsen kann.

3.8 Einleitung einer Gefahrenbremsung

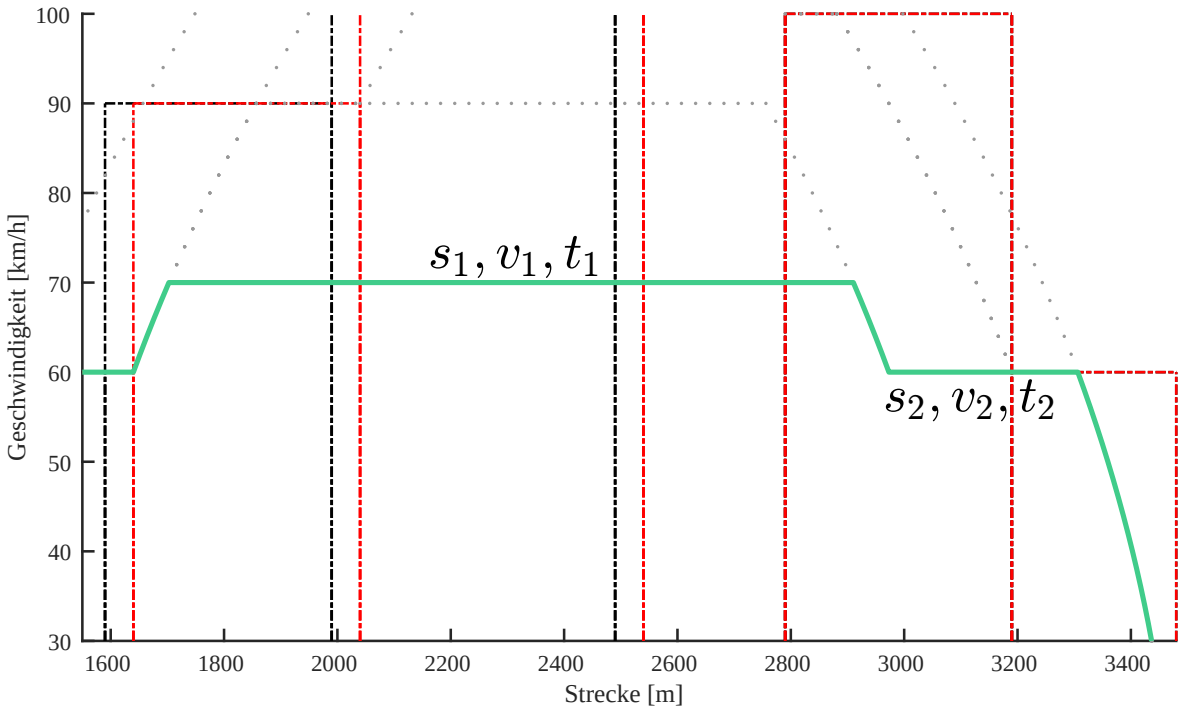

Abbildung 12: speedFineTuning_1

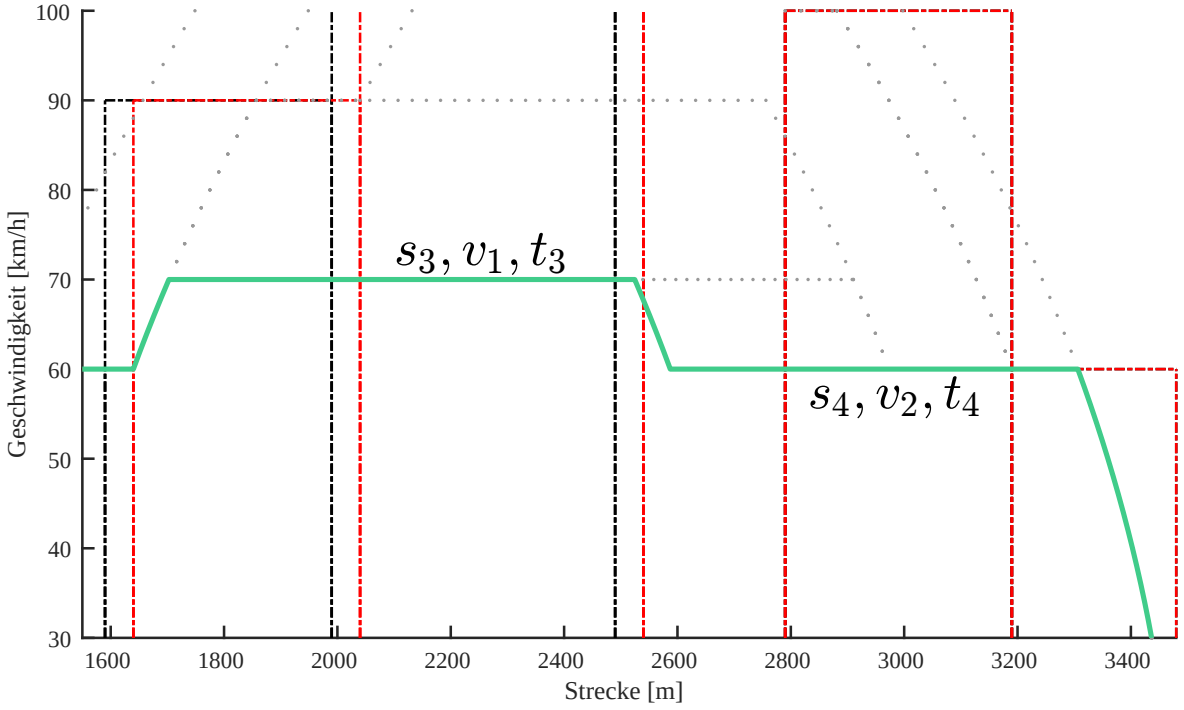

Abbildung 13: speedFineTuning_2

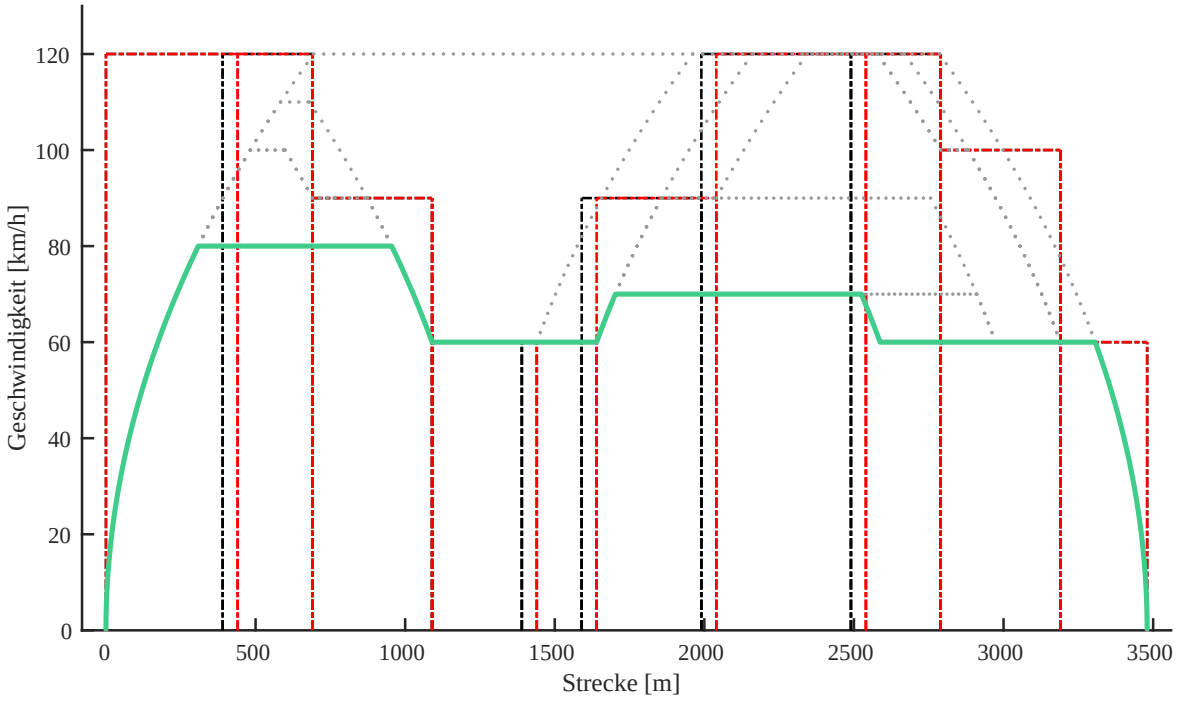

Abbildung 14: Finaler Fahrtverlauf

4 Formeln

4.1 Weg-Zeit Berechnung

Für die im folgenden Abschnitt verwendeten Gleichungen gilt:

$$
\begin{aligned}
a &= \text{Bremsverzoegerung } [m/s^2] \\
v &= \text{Geschwindigkeit } [m/s] \\
s &= \text{Strecke } [m] \\
t &= \text{Zeit } [s]
\end{aligned}
$$

Bei einer konstanten Beschleunigung gilt:

$$a(t) = a \tag{1}$$

Für die Bestimmung der Geschwindigkeit in Abhängigkeit der Zeit, muss die Beschleunigung $a(t)$ nach der Zeit t integriert werden.[5]

$$v(t) = \int a(t)\,dt \tag{2}$$

Daraus ergibt sich folgende Gleichung für die Geschwindigkeit in Abhängigkeit der Zeit. Die bei der Integration entstehende Integrationskonstante v_0 gibt dabei die Startgeschwindigkeit an.

$$v(t) = a \cdot t + v_0 \tag{3}$$

[5] Richard & Sander (2011, S. 20)

Für die Bestimmung der benötigten Zeit muss die Geschwindigkeit erneut integriert werden.[6] Die dabei entstehende Integrationskonstante s_0 gibt dabei die bereits zurückgelegte Strecke an.

$$s(t) = \int v(t)\,dt \tag{4}$$

$$s(t) = \frac{1}{2} \cdot a \cdot t^2 + v_0 \cdot t + s_0 \tag{5}$$

Bei der Verwendung dieser Gleichung werden die Integrationskonstanten v_0 und s_0 gleich 0 gesetzt, damit die Gleichungen allgemein gültig sind. Für die Berechnung des Beschleunigungs- und Abbremsverhalten der Fahrzeuge ist es notwendig zu wissen, welche Strecke ein Fahrzeug zurücklegen muss, um von einer Startgeschwindigkeit v_0 auf eine Zielgeschwindigkeit v_1 zu beschleunigen bzw. abzubremsen. Dafür wird die Gleichung für die Geschwindigkeit $v(t)$ nach $t(v)$ umgestellt und und in die Gleichung $s(t)$ eingesetzt. Daraus ergibt sich folgende Gleichung für die Strecke in Abhängigkeit von der Geschwindigkeit:

$$t(v) = \frac{v}{a} \tag{6}$$

$$s(v) = \frac{1}{2} \cdot \frac{v^2}{a} \tag{7}$$

Durch die Festlegung von $v_0 = 0$ wird so die benötigte Strecke ermittelt, welche ein Fahrzeug bei einer gegebenen Bremsverzögerung a benötigt, um von 0 m/s auf eine gegebenen Zielgeschwindigkeit v_1 zu beschleunigen. Bei der Berechnung des Beschleunigungs- und Abbremsverhalten wird es aber auch zu Situationen kommen, bei denen ein Fahrzeug eine Startgeschwindigkeit hat, für die gilt $v_0 \neq 0$. Um eine allgemein gültige Gleichung aufzustellen, wird für die Ermittlung der benötigten Strecke bei einer gegebenen Start- und Zielgeschwindigkeit die Strecke berechnet, die das Fahrzeug benötigt um von 0 m/s auf v_1 zu beschleunigen und von 0 m/s auf v_0. Für die gesuchte Strecke gilt dann:

$$s(v_0, v_1) = s(v_1) - s(v_0) \tag{8}$$

$$s(v_0, v_1) = \frac{1}{2} \cdot \frac{v_1^2 - v_0^2}{a} \tag{9}$$

In dem Programm übernimmt diese Berechnung die Funktion *getBrakeDistance()*. Damit keine negativen Rückgabewerte entstehen, wird im Falle einer Bremsung ($v_1 < v_0$) das Ergebnis mit -1 multipliziert. Beispiel eines Querverweis (9).

```
1 function getBrakeDistance (float $v_0, float $v_1, float $verzoegerung) {
2   if ($v_0 > $v_1) {
3     return $bremsweg = 0.5 * ((pow($v_0/3.6,2)-pow($v_1/3.6, 2))/($verzoegerung));
4   } else if ($v_0 < $v_1) {
5     return $bremsweg = -0.5 * ((pow($v_0/3.6,2)-pow($v_1/3.6, 2))/($verzoegerung));
6   } else {
7     return 0;
8   }
9 }
```

[6] ebd. (S. 20)

Neben der Berechnung der Strecke ist auch die benötigte Zeit essenziell. Dafür wird mittels $t(v)$ die Zeit berechnet, die das Fahrzeug benötigt, um von v_0 auf v_1 zu beschleunigen bzw. abzubremsen und aus der Differenz die benötigte Zeit berechnet.

$$t(v_0, v_1) = \frac{v_1 - v_0}{a} \tag{10}$$

In dem Programm übernimmt diese Berechnung die Funktion *getBrakeTime()*. Damit keine negativen Rückgabewerte entstehen, wird im Falle einer Bremsung ($v_1 < v_0$) das Ergebnis mit -1 multipliziert.

```
1 function getBrakeTime (float $v_0, float $v_1, float $verzoegerung) {
2   if ($v_0 < $v_1) {
3     return (($v_1/3.6)/$verzoegerung) - (($v_0/3.6)/$verzoegerung);
4   } else if ($v_0 > $v_1) {
5     return (($v_0/3.6)/$verzoegerung) - (($v_1/3.6)/$verzoegerung);
6   } else {
7     return 0;
8   }
9 }
```

Für die Berechnung einer Gefahrenbremsung ist es notwendig zu wissen, welche Geschwindigkeit das Fahrzeug an der Stelle der Gefahrenstelle hat. Dafür wird die Gleichung (9) nach v_2 umgestellt.

$$v_2(v_1, s) = \sqrt{-2 \cdot s \cdot a} + v_1 \tag{11}$$

4.2 Teil 2

4.3 speedFineTuning

Es gilt:

$$t_{ges} = t_1 + t_2 \tag{12}$$

$$s_{ges} = s_1 + s_2 \tag{13}$$

$$s = v \cdot t \tag{14}$$

Durch das Einsetzen der Gleichung (14) in die Gleichung (13) erhält man folgende Gleichung:

$$s_{ges} = v_1 \cdot t_1 + v_2 \cdot t_2 \tag{15}$$

Durch das Umstellen der Gleichung (12) nach t_2 und dem Einsetzen in Gleichung (15) gilt für t_1:

$$t_1 = \frac{s_{ges} - v_2 \cdot t_{ges}}{v_1 - v_2} \tag{16}$$

A Anhang

A.1 main.php

```php
<?php
// TODO:
// - Rename TimeDifference ($newTimeDifference)
// - Train Errors hinzufügen

// Load all required external files
require 'vorbelegung.php';
require 'functions/sort_functions.php';
require 'functions/cache_functions.php';
require 'functions/cache_functions_own.php';
require 'functions/ebuef_functions.php';
require 'functions/fahrtverlauf_functions.php';
require 'define_multicast.php';
require 'globalVariables.php';

// Set timezone
date_default_timezone_set("Europe/Berlin");

// Set memory
//ini_set('memory_limit', '1024M');

// Reports only errors
error_reporting(1);

// Define own train errors
$trainErrors = array();
$trainErrors[0] = "Zug stand falsch herum und war zu lang um die Richtung zu ändern.";
$trainErrors[1] = "In der Datenbank ist für den Zug keine Zuglänge angegeben.";
$trainErrors[2] = "In der Datenbank ist für den Zug keine v_max angegeben.";
$trainErrors[3] = "Zug musste eine Notbremsung durchführen.";

// Load static data from the databse into the cache
$cacheInfranachbarn = createCacheInfranachbarn();
$cacheInfradaten = createCacheInfradaten();
$cacheSignaldaten = createCacheSignaldaten();
$cacheInfraLaenge = createcacheInfraLaenge();
$cacheHaltepunkte = createCacheHaltepunkte();
$cacheZwischenhaltepunkte = createChacheZwischenhaltepunkte();
$cacheInfraToGbt = createCacheInfraToGbt();
$cacheGbtToInfra = createCacheGbtToInfra();
$cacheFmaToInfra = createCacheFmaToInfra();
$cacheInfraToFma = array_flip($cacheFmaToInfra);
$cacheFahrplanSession = createCacheFahrplanSession();
$cacheSignalIDToBetriebsstelle = createCacheToBetriebsstelle();
$cacheAdresseToID = array();  // Filled with data in getAllTrains()
$cacheIDToAdresse = array();  // Filled with data in getAllTrains()

// Global variables
$allTrainsOnTheTrack = array();  // All adresses found on the tracks
```

```php
$allTrains = array();    // All trains with the status 1 or 2
$allUsedTrains = array();   // All trains with the status 1 or 2 that are standing on the tracks
$allTimes = array();
$lastMaxSpeedForInfraAndDir = array();

// Get simulation and real time
$simulationStartTimeToday = getUhrzeit(getUhrzeit($cacheFahrplanSession->sim_startzeit, "simulationszeit", null, array("outputtyp"=>"h:i:s")), "simulationszeit", null, array("inputtyp"=>"h:i:s"));
$simulationEndTimeToday = getUhrzeit(getUhrzeit($cacheFahrplanSession->sim_endzeit, "simulationszeit", null, array("outputtyp"=>"h:i:s")), "simulationszeit", null, array("inputtyp"=>"h:i:s"));
$simulationDuration = $cacheFahrplanSession->sim_endzeit - $cacheFahrplanSession->sim_startzeit;
$realStartTime = time();
$realEndTime = $realStartTime + $simulationDuration;
$timeDifference = $simulationStartTimeToday - $realStartTime;

// Start Message
startMessage();

// Load all trains
// TODO: Funktion benötigt, die die Daten updatet...
$allTrains = getAllTrains();

// Loads all trains that are in the rail network and prepares everything for the start
findTrainsOnTheTracks();

// Checks if the trains are in the right direction and turns them if it is necessary and possible.
consoleCheckIfStartDirectionIsCorrect();
consoleAllTrainsPositionAndFahrplan();
showErrors();

// Adds all the stops of the trains.
addStopsectionsForTimetable();

// Adds an index (address) to the $allTimes array for each train.
initalFirstLiveData();

// Determination of the current routes of all trains.
calculateNextSections();

// Checks whether the trains are already at the first scheduled stop or not.
checkIfTrainReachedHaltepunkt();

// Checks whether the routes are set correctly.
checkIfFahrstrasseIsCorrrect();

// Calculate driving curve
calculateFahrverlauf();
$unusedTrains = array_keys($allTimes);
```

```php
$timeCheckAllTrainsInterval = 3;
$timeCheckAllTrains = $timeCheckAllTrainsInterval + microtime(true);
$timeCheckAllTrainErrorsInterval = 30;
$timeCheckAllTrainErrors = $timeCheckAllTrainErrorsInterval + microtime(true);
$sleeptime = 0.03;
while (true) {
  foreach ($allTimes as $timeIndex => $timeValue) {
    if (sizeof($timeValue) > 0) {
      $id = $timeValue[0]["id"];
      if ((microtime(true) + $timeDifference) > $timeValue[0]["live_time"]) {
        if ($timeValue[0]["live_is_speed_change"]) {
          if ($timeValue[0]["betriebsstelle"] == 'Notbremsung') {
            sendFahrzeugbefehl($timeValue[0]["id"], intval($timeValue[0]["live_speed"]));
            echo "Der Zug mit der Adresse ", $timeIndex, " leitet gerade eine
                ↪ Gefahrenbremsung ein und hat seine Geschwindigkeit auf ",
                ↪ $timeValue[0]["live_speed"], " km/h angepasst.\n";
          } else {
            sendFahrzeugbefehl($timeValue[0]["id"], intval($timeValue[0]["live_speed"]));
            echo "Der Zug mit der Adresse ", $timeIndex, " hat auf der Fahrt nach ",
                ↪ $timeValue[0]["betriebsstelle"],
            " seine Geschwindigkeit auf ", $timeValue[0]["live_speed"], " km/h
                ↪ angepasst.\n";
          }
        }

        $allUsedTrains[$id]["current_position"] =
            ↪ $timeValue[0]["live_relative_position"];
        $allUsedTrains[$id]["current_speed"] = $timeValue[0]["live_speed"];
        $allUsedTrains[$id]["current_section"] = $timeValue[0]["live_section"];

        if ($timeValue[0]["wendet"]) {
          //$allTimes[$timeIndex] = array();
          changeDirection($timeValue[0]["id"]);
        }

        if (isset($timeValue[0]["live_all_targets_reached"])) {
          $allUsedTrains[$id]["next_betriebsstellen_data"][$timeValue[0]["live_all_targets_reached"]]["ang
              ↪ = true;
          echo "Der Zug mit der Adresse ", $timeIndex, " hat den Halt ",
              ↪ $allUsedTrains[$id]["next_betriebsstellen_data"][$timeValue[0]["live_all_targets_reached"
              ↪ " erreicht.\n";
        }

        if ($timeValue[0]["live_target_reached"]) {

          $currentZugId = $allUsedTrains[$id]["zug_id"];
          $newZugId = getFahrzeugZugIds(array($id));

          if (sizeof($newZugId) == 0) {
            $newZugId = null;
          } else {
            $newZugId = getFahrzeugZugIds(array($timeValue[0]["id"]));
            $newZugId = $newZugId[array_key_first($newZugId)]["zug_id"];
```

```
141           }
142
143           // Fährt nach Fahrplan und hat keine neue Zug ID bekommen
144           if (!($currentZugId == $newZugId && $currentZugId != null)) {
145
146             if ($currentZugId != null && $newZugId != null) {
147               // neuer fahrplan
148               $allUsedTrains[$id]["operates_on_timetable"] = true;
149               getFahrplanAndPositionForOneTrain($id, $newZugId);
150               addStopsectionsForTimetable($id);
151               calculateNextSections($id);
152               //addNextStopForAllTrains($id);
153               checkIfFahrstrasseIsCorrrect($id);
154               calculateFahrverlauf($id);
155
156             } else if ($currentZugId == null && $newZugId != null) {
157               // fährt jetzt nach fahrplan
158               $allUsedTrains[$id]["operates_on_timetable"] = true;
159               getFahrplanAndPositionForOneTrain($id);
160               addStopsectionsForTimetable($id);
161               calculateNextSections($id);
162               //addNextStopForAllTrains($id);
163               checkIfFahrstrasseIsCorrrect($id);
164               calculateFahrverlauf($id);
165
166             } else if ($currentZugId != null && $newZugId == null) {
167               // fährt jetzt auf freier strecke
168               $allUsedTrains[$id]["operates_on_timetable"] = false;
169               calculateNextSections($id);
170               calculateFahrverlauf($id);
171             }
172           }
173         }
174         if (sizeof($timeValue) == 1 && !in_array($timeIndex, $unusedTrains)) {
175           array_push($unusedTrains, $timeIndex);
176         }
177         array_shift($allTimes[$timeIndex]);
178       }
179     }
180   }
181
182   if (microtime(true) > $timeCheckAllTrains) {
183     //var_dump($allUsedTrains[65]["current_section"]);
184     foreach ($unusedTrains as $unusedTrainsIndex => $unusedTrainsValue) {
185       $id = $cacheAdresseToID[$unusedTrainsValue];
186       compareTwoNaechsteAbschnitte($id);
187     }
188     // Search new trains.
189     // added to allUsedTrains and to unusedTrains
190     $prevTrains = array_keys($allUsedTrains);
191     findTrainsOnTheTracks();
192     $nextTrains = array_keys($allUsedTrains);
193     if (sizeof($prevTrains) != sizeof($nextTrains)) {
```

```
194       echo "Neu hinzugefügte Züge:\n\n";
195     }
196     foreach ($nextTrains as $nextTrainID) {
197       if (!in_array($nextTrainID, $prevTrains)) {
198         consoleCheckIfStartDirectionIsCorrect($nextTrainID);
199         consoleAllTrainsPositionAndFahrplan($nextTrainID);
200         addStopsectionsForTimetable($nextTrainID);
201         initalFirstLiveData($nextTrainID);
202         calculateNextSections($nextTrainID);
203         checkIfTrainReachedHaltepunkt($nextTrainID);
204         checkIfFahrstrasseIsCorrrect($nextTrainID);
205         calculateFahrverlauf($nextTrainID);
206         array_push($unusedTrains, $allUsedTrains[$nextTrainID]["adresse"]);
207       }
208     }
209     $timeCheckAllTrains = $timeCheckAllTrains + $timeCheckAllTrainsInterval;
210   }
211   if (microtime(true) > $timeCheckAllTrainErrors) {
212     foreach ($allUsedTrains as $trainKey => $trainValue) {
213       if (sizeof($trainValue["error"]) != 0 && !in_array(3, $trainValue["error"])) {
214         //var_dump($trainValue["id"]);
215       }
216     }
217     $timeCheckAllTrainErrors = $timeCheckAllTrainErrors +
          ↪ $timeCheckAllTrainErrorsInterval;
218   }
219   sleep($sleeptime);
220 }
```

A.2 globalVariables.php

```
1  <?php
2
3  $globalNotverzoegerung = 2; // Bremsverzögerung bei einer Notbremsung
4  $globalMinSpeed = 10; // Maximale Geschwindigkeit, wenn keine vorgegeben ist
5  $globalSpeedInCurrentSection = 60; // Maximale Geschwindigkeit im aktuellen Abschnitt
6  $globalFirstHaltMinTime = 20; // calculateFahrverlauf -> Zeit fürs Wenden...
7  $globalIndexBetriebsstelleFreieFahrt = 9999999999999;
8  $globalFloatingPointNumbersRoundingError = 0.0000000001;
9  $globalTimeOnOneSpeed = 20;
10
11 $useSpeedFineTuning = true;
12
13 $useMinTimeOnSpeed = true;
14 $errorMinTimeOnSpeed = false;
15
16 $slowDownIfTooEarly = true;
```

Literatur

Maschek, U. (2013). Zugbeeinflussung. In *Sicherung des schienenverkehrs* (S. 184–212). Springer.

Richard, H. & Sander, M. (2011). *Technische mechanik. dynamik: Grundlagen - effektiv und anwendungsnah.* Vieweg+Teubner Verlag.

Wende, D. (2013). *Fahrdynamik des schienenverkehrs.* Springer-Verlag.