

FH Aachen

**Fachbereich
Elektrotechnik und Informationstechnik**

Bachelorarbeit

**Erzeugung eines künstlichen Datensatzes zur Bremsprognose von
Güterzügen und Validierung der Big Data Verarbeitung**

**Simon Westfechtel
Matr.-Nr.: 3120254**

Referent: Prof. Dr-Ing. Ingo Elsen

Korreferent: Prof. Dr. Raphael Pfaff

August 9, 2020

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Aachen, August 9, 2020

Danksagung

Danke.

Contents

1	Introduction	11
1.1	Background	11
1.1.1	Railway Vehicle Operations	11
1.1.2	Automatic Train Protection and Cab Signaling	12
1.1.3	Braking Curves	13
1.2	Problem	13
1.3	Solution	14
2	Fundamentals of Railway Vehicle Engineering	15
2.1	The Train Brake	15
2.2	Influences on Braking Performance	16
2.2.1	Inherent factors, model variables	16
2.2.2	Relations	18
2.3	Constants	20
2.4	Time considerations	20
3	Modeling of Train Operations	23
3.1	Matlab Simulink	23
3.2	Initial Model	23
3.3	Model Expansion	25
3.4	Outlook	33
4	Data Generation	35
4.1	Simulation	35
4.2	Matlab Code	36
4.3	Data Structure	41
4.4	Analysis of generated Data	43
4.5	Performance	43
5	Conclusion	45
	Bibliography	47
	List of Figures	47

List of Tables	49
A Data visualization	53

Abstract Modern day railway system operations require automated train control mechanisms, e.g. European Train Control System ETCS, to maximize efficiency, which is often times limited by outdated infrastructure, as well as safety of operations. One way to achieve this is by lowering the required distance between two trains on the same track, which in turn demands a reliable method of predicting the braking distance at any given moment.

While determination of the necessary braking curves is feasible for a limited number of train formations, the large diversity of vehicles in freight operations poses an issue. One approach for a solution would be using Big Data, which would be able to process the required amounts of data to calculate reliable braking curves even for freight operations.

The problem here is that there is simply not enough data available since freight trains usually don't have the sensory equipment needed. To circumvent that obstacle, this work proposes generation of artificial data via white box modeling to be then used in further big data operations.

Chapter 1

Introduction

Introduction This section describes the background and motivation of the research (section 1.1), the problem to be addressed (section 1.2) and the proposed solution (section 1.3).

1.1 Background

In recent years, data science has become more and more prominent. Since large quantities of data have become omnipresent, big data processing is applicable in various fields of research and operations. This work focuses on the application of big data processing to railway system operations, more specifically the preconditions. Mainly, two areas show promise to benefit from big data usage, which is predictive maintenance as well as optimization of railway operations. As the name suggests, lots of data is required, think hundreds of terabyte for freight operations in Germany alone. As it stands though, unfortunately, this data source remains largely untapped today, owing to freight wagons typically lacking necessary sensory equipment. This is where this work comes into play.

1.1.1 Railway Vehicle Operations

Road and rail traffic are fundamentally different, mainly in two regards. First, the physical properties, which will be discussed in chapter 2, second, the actual modalities of operations, which will be topic of this section.

In road traffic, there are many different vehicles, all operating independently from one another. Although there is ongoing experimentation to utilize autonomous vehicles to recreate train-like lorry configurations, this is not the norm. In contrast to that, the track guiding of wheels offered by rails enables the formation of trains possibly kilometers long, reducing labor cost, infrastructure usage and energy consumption. This, of course, calls for special safety measures to be taken, since higher speeds and payloads result in long braking distances.

Figure 1.1 illustrates the three main principles of spacing between two trains on the same track. $s_{b,i}$ denotes the braking distance of train i , $l_{t,i}$ the length of train i , l_{bl} the length of a track section, and S a safety margin. In fixed spatial distance, the distance between two trains must at least be the maximum braking distance $S_{b,max}$, plus the length of one track section (i.e. between two signals) l_{bl} , plus a safety margin S . In absolute braking distance, the distance between two trains must at least be the braking distance of the second train $S_{b,2}$, plus

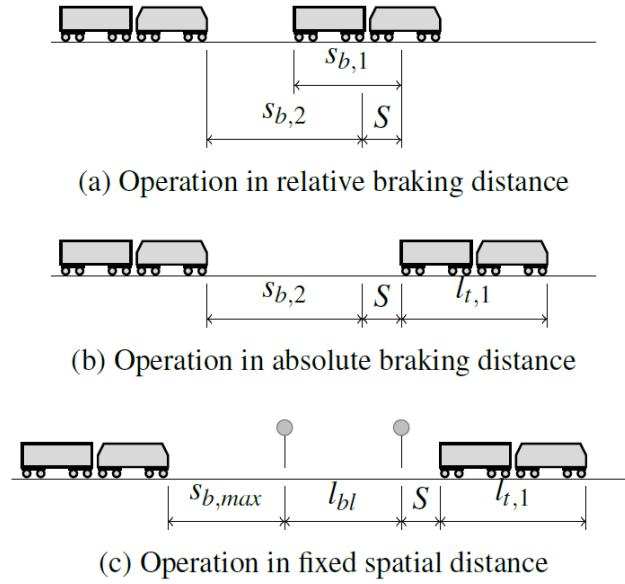


Figure 1.1: Spacing paradigms [Pfa17b]

a safety margin S . Finally, in relative braking distance, the distance between two trains must be sufficient for the second train to be able to come to a full halt behind the first train if both trains start braking at the same time. Fixed spatial distance, the most commonly used today, is very inefficient in terms of track utilization, but offers the most protection against accidents. In order to optimize efficiency, it is desirable to move towards relative braking distances, which requires the ability to very accurately predict braking capacity of trains.

1.1.2 Automatic Train Protection and Cab Signaling

Automatic train protection systems are designed to ensure safe operation in case of human or technical error or malpractice by the train operator. Generally, all trains operate on track sections which are free of other vehicles, reserved and locked. Think for example a section between two signals. These sections are also referred to as movement authority, and a train has to be capable of coming to a complete halt before the end of movement authority as to not violate a track section locked by another train. For this, the driver and/or the onboard computer needs to be informed about the endpoint of the current movement authority as well as speed limits. The train protection system enforces application of brakes in case of violation of restrictions, for example if the train exceeds the speed limit or the train would otherwise be unable to stop in time before movement authority expires. Train protection systems may be categorized by means of information transfer. Spatially discrete acting systems use transmitters placed at strategic points along the track, for example signals, whereas continuous systems may transmit information at all times, either via track-sided wire loops or radio communication. These allow higher degrees of automation than discrete systems, as trains have access to the necessary real-time data.

Cab signaling relays all relevant information to the train operators so they may act in the most optimal way. The European train control system, short ETCS, encompasses both of these ([Hav17], p.12).

1.1.3 Braking Curves

For the ETCS to be able to supervise train velocity, it needs to determine the vehicle's braking capacity for any given moment in time, using a mathematical model of the braking dynamics and of the track characteristics. This prediction is called a braking curve, which describes the probable braking process for a set of input parameters. For an example, refer to figure 1.2. It shows a braking process, in particular the decline of the train's velocity over distance. Since braking performance cannot be predicted with absolute certainty, an ϵ parameter needs to be factored in to account for random behavior. Therefore, there is a number of discrete braking curves for any set of parameters, forming a probability distribution [Pfa17b]. The parameters may be classified in four categories:

- Physical parameters, which are determined by real time measurements from on-board equipment
- Fixed values, like driver reaction time
- Trackside data, like track gradient or signaling data
- Train data, mostly captured beforehand, relating to the rolling stock braking system itself

The fourth category, train data, is the key factor for this work, as will be explained below.

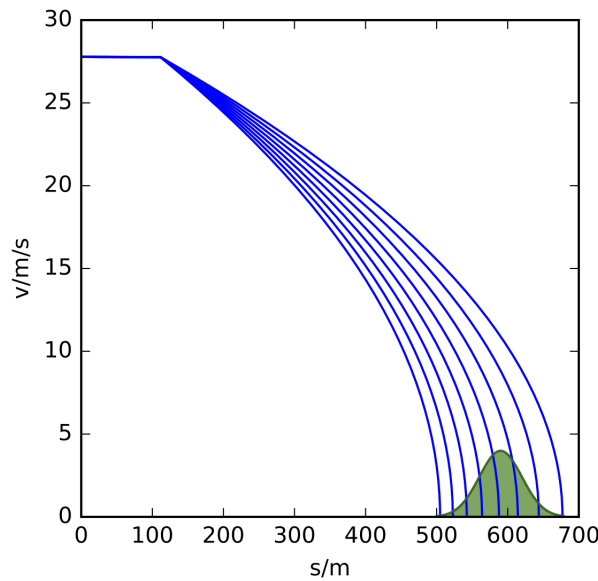


Figure 1.2: Braking curves [Pfa17b]

1.2 Problem

As mentioned, one category of input parameters for the calculation of braking curves is data directly associated with the rolling stock, and in particular to the braking system itself. While this does not necessarily present itself as an issue in passenger trains, freight wagons, in contrast,

are usually not electrified and therefore do not currently possess the sensory equipment that would be necessary to obtain such data in an adequate quantity and quality, especially in regards to *big data processing*. Although it has been proposed to equip freight wagons accordingly (Refer to the concept “freight wagon 4.0” by Dr. Manfred Enning and Dr. Raphael Pfaff [EP17]), it will be years, possibly decades, before enough rolling stock has been retrofitted as to make it possible to obtain the desired data. Since braking curves vary according to the rolling stock, this lack of data makes calculation thereof impossible.

1.3 Solution

Since the problem is lack of data from real life operations, this work proposes to circumvent this by generation of artificial, simulated data instead. The data set should replicate the actual distribution of braking behavior as closely as possible. It is therefore necessary to first create a model encompassing the braking process of a freight train. This model will be discussed in depth in chapter 3. By using that model to run a large number of simulated braking processes, one may obtain data about the behavior of the braking system, which, albeit being artificial, should at least satisfy requirements for the calculation of rudimentary braking curves. A freely configurable simulation environment further allows for great flexibility in terms of input parameters, therefore enabling for covering a very large range of data, where computing power and time are the only limiting factors. The process of data generation and simulation will be discussed further in chapter 4.

As real life operations would yield very high quantities of data, the simulation output must be stored in a data structure which is suitable for big data processing. This structure will also be discussed in chapter 4.

Chapter 2

Fundamentals of Railway Vehicle Engineering

Introduction This section deals with the engineering backgrounds of the work, especially in regards to railway vehicle engineering and railway physics. For the creation of a braking model, it is necessary to have at least a basic understanding of the engineering and physics behind rail traffic. The most fundamental component to understand is, of course, the actual braking process.

2.1 The Train Brake

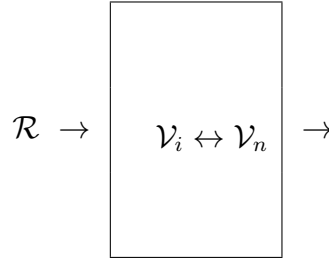
There are a number of different construction methods for brakes, but each must address two problems: The means of transmission of commands and the actual form of brake force generation. The focus here will lie on the pneumatic brake, since it is still the most prevalent in cargo vehicles. From a macro perspective, the pneumatic brake has three main components: The train driver's brake valve, the brake pipe and one or more brake cylinders for each wagon or locomotive. The driver's valve and the brake pipe are primarily the means of command transmission, while the brake cylinder's job is the generation of brake force.

The braking system is an indirect one, which means the brake pipe needs to be vented to apply the brakes. Therefore, the brake pipe is kept at a running pressure of 5 bar. One advantage of this approach is that brakes are automatically applied in case of train separation. The downside, however, is that command transmission is limited to the speed of sound. In order to generate that pressure, the locomotive contains a compressor and an air reservoir. Initially, the brake pipe as well as all auxiliary air reservoirs are brought to operating pressure. If the train operator wishes to brake, he has to actuate the driver's brake valve, which in turn vents the brake pipe, usually as far as 3.5 bar (full braking), sometimes even lower. As rule of thumb, the more pressure is vented, the more braking force is applied. Each wagon brake has, in essence, an auxiliary air reservoir, a distributor valve and a braking cylinder. When the brake pipe is vented, the air in the auxiliary reservoir has a higher pressure. This difference in pressure triggers a switch of the distributor valve, and the air from the reservoir may go into the brake cylinder. This increase in the cylinder generates the actual braking force. To summarize, the amount of generated braking pressure is directly proportional to the amount

of pressure vented.

2.2 Influences on Braking Performance

Since the aim of this work is to monitor braking performance, it becomes necessary to understand which factors influence the braking process, and how. The braking process is, in its core, a system. A system may be defined as a set of objects, which are interconnected by relations. It is enclosed by its environment, which may or may not affect the system itself. Below is an illustration, also called box model:



The box represents the actual system, \mathcal{V}_i are the system variables, or objects, their relations represented by the \leftrightarrow . The \mathcal{R} represents the system's environment and its influence on the system itself. Finally, the rightmost arrow \rightarrow is the influence the system might have on the environment, though this is often disregarded as the focus lies on what happens inside the system. Creating a box model for the braking process makes for an easier identification of the influencing factors, and allows for their categorization into inherent, i.e. system variables, and external, i.e. system environment.

From a modeling perspective, it is sensible to differentiate as follows: Everything related to the train itself is part of the system, and everything else is part of the system's environment. As per definition, it is compromised of a set of variables, a set of relations, and a set of constants. Let us make a formal definition of the model for the braking process system:

$$\mathcal{M} = \{\mathcal{V}, \mathcal{R}, \mathcal{C}\}$$

where \mathcal{V} is the set of system variables, \mathcal{R} is the set of relations (a) between elements of \mathcal{V} and (b) between system and environment, and \mathcal{C} is the set of system constants. The system variables may be subsystems themselves. We shall take a closer look at \mathcal{V} first.

2.2.1 Inherent factors, model variables

Looking at \mathcal{M} , which is the system describing the braking process, \mathcal{V} consists of all factors inherent to the train which have an impact on the braking, as well as the relevant quantifiers. The main ones are:

- The train brake
- The wagon mass, denoted as m

- The train velocity, denoted as v
- The train composition, i.e. in which order the wagons are, for example the heaviest wagons being in front and the lighter ones at the back, denoted as $comp$
- The generated braking force, denoted as F_b
- The train's deceleration, denoted as a
- The train's braking distance, denoted as d

It is noteworthy that the train brake is another system in itself, which is perfectly fine of course. Alas, it is therefore necessary to define another model.

$$\mathcal{M}_B = \{\mathcal{V}_B, \mathcal{R}_B, \mathcal{C}_B\}$$

Let \mathcal{M}_B be the model describing the system train brake, with $\mathcal{V}_B = \{\mathcal{M}_{bv}, \mathcal{M}_{bp}, \mathcal{M}_{bc}\}$, where \mathcal{M}_{bv} is the model of the brake valve, \mathcal{M}_{bp} the model of the brake pipe and \mathcal{M}_{bc} the model of the brake cylinder, which in turn shall be defined as follows:

$$\mathcal{M}_{bv} = \{\mathcal{V}_{bv}, \mathcal{R}_{bv}, \mathcal{C}_{bv}\}$$

Let \mathcal{M}_{bv} be the model describing the system brake valve, with $\mathcal{V}_{bv} = \{x\}$, $\mathcal{R}_{bv} = \emptyset$ and $\mathcal{C}_{bv} = \emptyset$, where x is the state of the brake valve, i.e. its opening percentage, ranging from 0 (fully closed) to 1 (fully open, full braking).

$$\mathcal{M}_{bp} = \{\mathcal{V}_{bp}, \mathcal{R}_{bp}, \mathcal{C}_{bp}\}$$

Let \mathcal{M}_{bp} be the model describing the system brake pipe, with $\mathcal{V}_{bp} = \{l_{bp}, p_{bp}\}$, $\mathcal{R}_{bp} = \emptyset$ and $\mathcal{C}_{bp} = \{v_{bp}\}$, where l_{bp} is the physical length of the brake pipe, p_{bp} is the pressure on the brake pipe and v_{bp} is the propagation velocity of the pipe's medium.

$$\mathcal{M}_{bc} = \{\mathcal{V}_{bc}, \mathcal{R}_{bc}, \mathcal{C}_{bc}\}$$

Let \mathcal{M}_{bc} be the model describing the system brake cylinder, with $\mathcal{V}_{bc} = \{p_{bc}\}$, $\mathcal{R}_{bc} = \emptyset$ and $\mathcal{C}_{bc} = \{t_{bc}\}$, where p_{bc} is the cylinder's pressure and t_{bc} is the cylinder's fill time.

This wraps up the definition of \mathcal{V}_B . We may now get to the definition of \mathcal{R}_B , which consists of the relations (a) between the elements of \mathcal{V}_B and (b) between \mathcal{M}_B and its environment. Let us begin with (a):

Convention For readability purposes, \propto shall from here on denote direct proportionality, and \sim shall denote inverse proportionality.

\mathcal{M}_{bv} relates to \mathcal{M}_{bp} in the sense that the state of the brake valve has a direct influence on the pressure on the brake pipe. More specifically, $x \in \mathcal{V}_{bv}$ is inversely proportional to $p_{bp} \in \mathcal{V}_{bp}$. The higher the value of x , i.e. the opening percentage of the brake valve, the lower the value of p_{bp} , i.e. the pressure on the brake pipe. We can therefore denote $x \sim p_{bp}$, and subsequently define a relation $R_{bv,bp} : \mathcal{M}_{bv} \sim \mathcal{M}_{bp}$.

\mathcal{M}_{bp} relates to \mathcal{M}_{bc} in the sense that the pressure on the brake pipe has a direct influence on the brake cylinder's pressure. More specifically, $p_{bp} \in \mathcal{V}_{bp}$ is inversely proportional to $p_{bc} \in \mathcal{V}_{bc}$. The lower the value of p_{bp} , i.e. the pressure on the brake pipe, the higher the value p_{bc} , i.e. the pressure in the brake cylinder. We can therefore denote $p_{bp} \sim p_{bc}$, and subsequently define a relation $R_{bp,bc} : \mathcal{M}_{bp} \sim \mathcal{M}_{bc}$.

For simplicity, we can look at \mathcal{M}_B as being isolated from its environment, i.e. the environment does not impact \mathcal{M}_B , and vice versa. Therefore, (b) is void. Consequently, we can define \mathcal{R}_B as follows:

$$\mathcal{R}_B = \{R_{bv,bp}, R_{bp,bc}\}$$

This finally leaves us with c_B . Since call constants are specific to the respective sub-models $\in \mathcal{V}_B$, \mathcal{M}_B has no constants of its own, therefore $c_B = \emptyset$. This completes the definition of \mathcal{M}_B , and we can properly define the set of model variables of the model braking process:

$$\mathcal{V} = \{\mathcal{M}_B, m, v, comp, F_b, a, d\}$$

2.2.2 Relations

Looking at our model \mathcal{M} , we have now defined its set of variables, \mathcal{V} . Next, it is necessary to define its set of relations, \mathcal{R} , which may be distinguished into internal and external relations.

Relations between system variables

Obviously, the train brake \mathcal{M}_B relates to F_B as it is the actor responsible for generating the braking force. More specifically, $p_{bc} \in \mathcal{V}_{bc}$, which is the pressure in the brake cylinder, is directly proportionate to F_b , meaning the higher the pressure, the more force is applied. We can therefore denote $p_{bc} \propto F_b$, and subsequently define a relation $R_{\mathcal{M}_B, F_b} : \mathcal{M}_B \propto F_b$.

Furthermore, the wagon mass m relates to F_B as well. The vehicle mass is proportional to the kinetic energy to dissipate during braking actions. The larger m is, the more kinetic energy ($E = \frac{1}{2} \cdot m \cdot v^2$), and thus the higher the required braking force to slow the train down. As the vehicle mass varies, so must the applied braking force in order to avoid wheel slip and locking of the wheels which occurs if too much force is applied ([Cru12], p.47). We can therefore denote $m \propto F_b$, and subsequently define a relation $R_{m, F_b} : m \propto F_b$.

The braking deceleration a , neglecting other resistances, is influenced by two factors: The applied braking force F_b , and the vehicle's mass m . As per Newton's second law of motion, $F = m \cdot a$, so $a = \frac{F_b}{m}$ [train braking, 23]. In practice, this means the braking force is directly proportional to the braking deceleration, and the vehicle mass is inversely proportional to a ,

not taking into consideration wheel slip. We can therefore denote $F_b \propto a$, and $m \sim a$, and subsequently define two relations $R_{F_b,a} : F_b \propto a$, and $R_{m,a} : m \sim a$.

Finally, the braking distance d is also dependent on two factors: The running speed v , and the braking deceleration a . We have

$$\begin{aligned} d &= \frac{1}{2} \cdot a \cdot t^2 \\ v &= a \cdot t \quad \rightarrow \quad d = \frac{v^2}{2 \cdot a} \end{aligned} \tag{2.1}$$

where t is the stopping time ([Cru12], p.51). So in practice, the square of the running speed v is directly proportional to the braking distance, and the braking deceleration is inversely proportional to d . We can therefore denote $v \propto d$, and $a \sim d$, and subsequently define two relations $R_{v,d} : v \propto d$, and $R_{a,d} : a \sim d$.

This completes the set of internal relations of \mathcal{R} .

Relations between environment and system

To recapitulate, \mathcal{M} is a model for the braking process. Having handled its internal relations, it is now time to focus on the relation between the system and its environment. As established in section 2.2, the environment consists of all factors which have an influence on the braking process, but are not inherent to the train itself. In practice, this is the track which the train operates on. More specifically, it is two main properties of the track:

- The track gradient, denoted as α
- The wheel/rail friction coefficient, denoted as μ

As shown earlier, the braking deceleration has a substantial influence on the braking process, namely on the braking distance and on the stopping time (see equation 2.1). So far, the deceleration was only related to vehicle mass m and applied braking force F_b , both being internal relations. However, the track gradient, also has an influence on a . As the train goes uphill, the deceleration increases; as it goes downhill, it decreases, due to gravity. A simple formula might look like this:

$$a = a_{brake} + a_{grad} \tag{2.2}$$

where a is the total braking deceleration, a_{brake} is the deceleration resulting from the application of the brakes, and a_{grad} is the acceleration/deceleration resulting from the track gradient. Let a_{grad} be positive for upward, negative for downward slopes. We can define a_{grad} as a step function

$$a_{grad}(\alpha) = \begin{cases} \geq 0 & \text{if } \alpha \geq 0 \\ < 0 & \text{if } \alpha < 0 \end{cases} \tag{2.3}$$

where α is the value of the gradient, e.g. a percentage or an angle. This shows the track gradient is directly proportional to the total braking deceleration. We can therefore denote $\alpha \propto a$, and subsequently define a relation $R_{\alpha,a} : \alpha \propto a$.

The second factor to look at is the wheel/rail adhesion. Environmental effects like rain, ice or contaminants such as leaves have a degrading effect on the wheel/rail friction. This is expressed by a friction coefficient μ , where a lower value, for example $\mu = 0.2$ for a wet rail surface, means worse braking performance compared to a higher value, e.g. $\mu = 0.4$ for normal conditions. The friction coefficient is a limiting factor for the applied braking force, in the sense that when too much force is applied, wheel slip occurs, and braking performance decreases ([BGS06], p.876). While μ does not have a direct influence on F_b , it is, in practice, still related to it in that it is a limiting factor; an indirect influence, so to speak. In that regard, we might say it is directly proportional to the braking force: The higher the friction coefficient, the higher the maximum braking force that may be applied. We can therefore denote $\mu \propto F_b$, and subsequently define a relation $R_{\mu, F_b} : \mu \propto F_b$.

We may now properly define the set of relations of the model braking process:

$$\mathcal{R} = \{R_{\mathcal{M}_B, F_b}, R_{m, F_b}, R_{F_b, a}, R_{m, a}, R_{v, d}, R_{a, d}, R_{\alpha, a}, R_{\mu, F_b}\}$$

2.3 Constants

The braking process, being a dynamic one, has no constants in itself. Some of its components do, in particular the models making up the train brake, like the brake pipe \mathcal{M}_{bp} 's propagation velocity v_{bp} . However, these have already been discussed; we can therefore conclude that $\mathcal{C} = \emptyset$.

To summarize, we have now defined all necessary components of \mathcal{M} , namely its set of model variables \mathcal{V} , its set of relations \mathcal{R} , and its set of constants \mathcal{C} . This model will be used in chapter 3 as a theoretical basis upon which to build a Simulink model, simulating the braking process.

2.4 Time considerations

So far, we have looked at the braking process from a static point of view. However, in reality it is a dynamic process and thus, time has to be taken into consideration. More specifically, it is a continuous model: For \mathcal{M} , the state of its variables changes all the time for the whole duration of the process.

Remember how we have defined the model. It is a set comprising of a number of subsets; but it describes a system, in this case the braking process of a train. This means that the model variables, i.e. the elements of \mathcal{V} , are subject to change as the process progresses. This can be expressed by another equation, in form of a function f :

$$\begin{aligned} \{V\} &= f(\mathcal{M}) \\ &= f(\mathcal{V}, \mathcal{R}, \mathcal{C}) \end{aligned}$$

This suffices to determine the system's state before and after a relation has changed. As an example, v might be $25 \frac{m}{s}$ when the train operator engages the brakes, and $10 \frac{m}{s}$ when the brakes have been released. However, since we wish to monitor the whole braking process, it is also interesting to see how the state changes over time. In this example, v would be $25 \frac{m}{s}$ for $t = 0$, $22.3 \frac{m}{s}$ for $t = 50$ etc. We therefore need a new equation where \mathcal{V} is also dependent on t .

$$\mathcal{V}(t)' = f(\mathcal{M})$$

This allows for the calculation of the rate of change for any given point in time. By integrating the equation, one can also obtain the actual values of the elements of \mathcal{V} , e.g. the value of v for $t = 120$. The function f is one, or a set of, differential equations. In practice, these are automatically generated and calculated by Simulink.

We have now laid all the necessary groundwork to build a functioning model of the braking process, which can then be used for simulation and data generation. The next chapter will deal with this.

Chapter 3

Modeling of Train Operations

Introduction Chapter 2 has established the theoretical model of the train braking process. It will serve as a foundation to build a practical model suitable for conducting actual simulations.

3.1 Matlab Simulink

The model has been built utilizing the software Matlab Simulink by Mathworks. It offers several distinct advantages: It uses a graphical block diagramming tool which makes it very intuitive to use. At the same time, various official and third-party add-on libraries make it very versatile. Most importantly, as it is tightly integrated with the Matlab environment, it can be used for both modeling and simulating dynamical systems, like the braking process of a train.

3.2 Initial Model

Looking back at chapter 2, the aim is to build a model for the braking process of a freight train. An initial model of a single braking procedure, kindly provided by Dr. Raphael Pfaff [Pfa19], will serve as a basis. Most of the components previously designed in chapter 2 can already be found therein.

Figure 3.1 shows a model of a freight train of fixed length, consisting of 40 wagons in total. For better readability, five wagons are condensed into one subsystem each, marked by the red rectangle. The subsystems are interconnected by a brake pipe, marked by the green arrow. They have one input port for the incoming pressure on the brake pipe, and three output ports, one for the pipe connection to the next subsystem, and two for recording brake pressure and brake force (marked by the pink and blue arrows, respectively). A depiction of a subsystem can be found in appendix A.1.

Figure 3.2 shows a model of a freight wagon. It already encompasses several of the model variables defined in section 2.2.1. The brake pipe \mathcal{M}_{bp} is represented by the green arrows. It leads from input port (pink rectangle) to output port (yellow rectangle); the propagation delay is realized by a transport delay block (green rectangle), using propagation velocity v_{bp} and wagon length l_{bp} . The brake pressure p_{bp} is the value of the signal traveling through the connection. Note that in the simulation, the signal value for p_{bp} is the difference between regular



The brake cylinder \mathcal{M}_{bc} is represented by the red rectangle. The pressure on the brake pipe p_{bp} is converted into a coefficient between zero and one, where zero equals no brakes applied, or $p_{bp} = 5$, and one equals full braking pressure, or $p_{bp} = 3.5$. The exact formula is this:

So for a braking pressure of 4 bar, the coefficient would be $(4 - 5) \cdot \frac{-1}{5-3.5} = -1 \cdot \frac{-1}{1.5} = \frac{2}{3}$. A

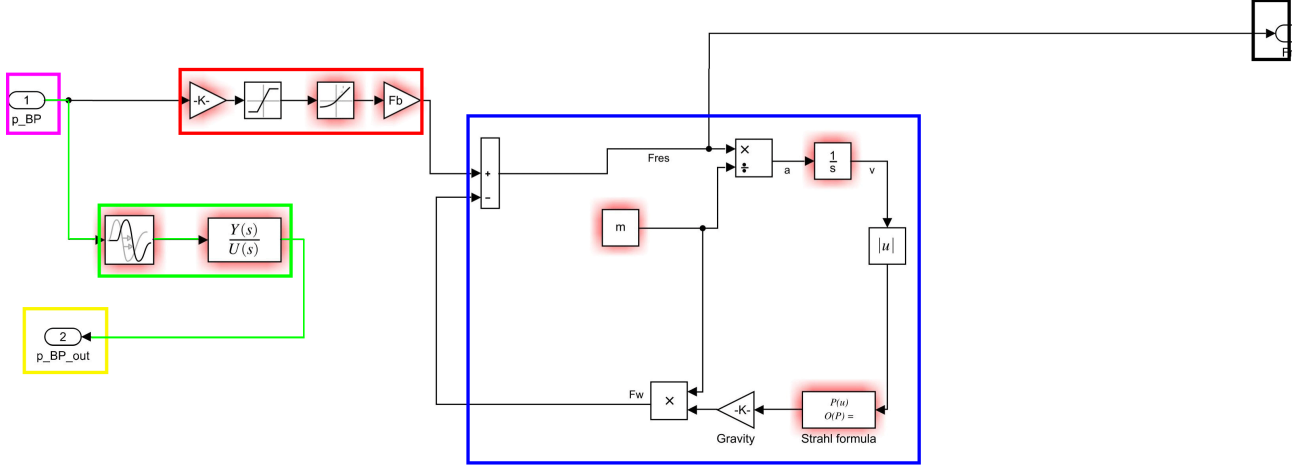


Figure 3.2: Initial Model - Wagon

rate limiter block accounts for the brake cylinder's filling time t_{bc} . Finally, the braking force is calculated by multiplying the maximum braking force, which is mainly dependent on the wagon mass m ([Cru12], p.50). So for the braking pressure, we have:

$$F_b = F_{b,max} \cdot p_{bc}$$

The final component is the factoring in of the driving resistance of the train, utilizing Strahl's formula ([Wen03], p.153). This is represented by the blue rectangle. By continuous-time integration of the acceleration, which is calculated according to Newton's second law of motion ($a = \frac{F}{m}$), it is possible to determine the velocity, which in turn allows for the calculation of the force generated by the driving resistance. This force is then added to the braking force and fed into an output port, marked with the black rectangle.

Regarding simulation, this initial model is only fit for one single braking process, where a train of fixed length and fixed composition, brakes from an initial velocity all the way to a halt; the only variation is the pressure on the brake pipe, which is also the sole input for the simulation. Appendix A.2 shows how that input might look like.

3.3 Model Expansion

This initial model is however not of sufficient detail. Where it merely simulates a single, rather undynamic braking process, the goal is the simulation of a train ride, with alternating phases of braking and accelerating. The model should furthermore factor in more components of the theoretical model \mathcal{M} discussed in chapter 2, as well as allow for the recording of more properties specific to the braking process. For that purpose, the model has to be expanded. The simulation input needs to be changed in order to allow for the simulation of a train ride. Previously, it being only one braking process, using braking pressure as input was the obvious choice. This will be replaced by a track profile, according to which the train either engages its brakes, or accelerates. This will be discussed in further detail in chapter 4.

In the initial model, the train is only able to decelerate by engaging its brakes. To allow for acceleration, it is necessary to expand the model accordingly. This may be realized by a rather simple two-point controller, which means the train is either accelerating or decelerating. The decision on whether to engage brakes or apply traction force at a particular point in time is made utilizing the track profile: If the current velocity of the train v_{real} is greater than the maximum allowed velocity v_{max} at that time, the train engages its brakes by lowering the pressure on the brake pipe p_{bp} . If v_{real} is less than v_{max} , the train applies a traction force F_t in order to increase speed. Both p_{bp} and F_t scale with the discrepancy between v_{real} and v_{max} . This means the higher the value of v_{dif} is, the more pressure is vented from the brake pipe (or the more traction force is applied). This prevents the train from overcompensating, for example applying full braking pressure to decrease velocity by $2 \frac{m}{s}$. Notice there is actually no case for when v_{real} equals v_{max} , i.e. $v_{dif} = 0$. One might expect that this would lead to the train constantly oscillating between accelerating and decelerating, but in practice this does not occur, and it is thus unnecessary to account for that case.

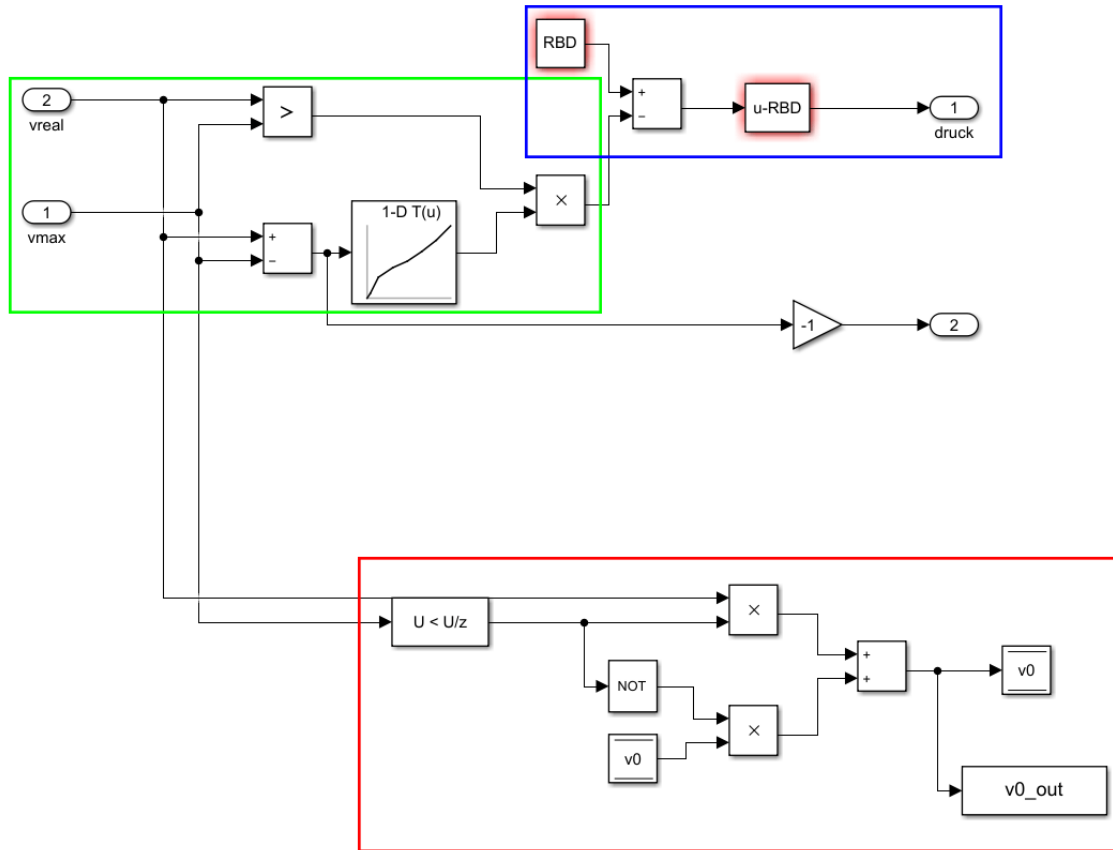


Figure 3.3: Expanded Model - Pressure Calculation

Figure 3.3 shows the subsystem responsible for the calculation of the braking pressure p_{bp} . The difference between v_{real} and v_{max} is used as input for a one-dimensional lookup table, which in

essence is a step-function mapping different input values to discrete output values.

$$H(n) = \begin{cases} 0.1 & \text{if } n = 1 \\ 0.7 & \text{if } n = 15 \\ 0.8 & \text{if } n = 20 \\ .. & \end{cases} \quad (3.1)$$

Equation 3.1 is an example for how this function might look like. The input parameter n stands for the discrepancy between actual and maximum velocity v_{dif} , and the output of the $H(n)$ represents the amount of pressure to vent from the brake pipe. This allows for adjusted braking, where p_{bp} is directly proportional to v_{dif} .

To account for the train only engaging its brakes in case its current velocity is higher than the maximum allowed velocity, a relational operator $>$ is used to compare v_{real} to v_{max} .

$$P(n, t) = H(n) \cdot (v_{real}(t) > v_{max}(t)) \quad (3.2)$$

Equation 3.2 describes the logic. The braking pressure for a velocity discrepancy n at a point in time t equals the result of $H(n)$ (equation 3.1) multiplied with the result of the relational operation comparing $v_{real}(t)$ to $v_{max}(t)$, which is 1 if $v_{real}(t)$ is greater than $v_{max}(t)$, and otherwise 0. This ensures that $P(n, t)$ is zero when the train's velocity is lower than the maximum allowed velocity, and thus no pressure is vented from the brake pipe, and no brakes are engaged. This part of the system is marked by the green rectangle in figure 3.3. The blue rectangle marks the conversion of the braking pressure into a signal fit for further calculations, analogous to the initial model, discussed previously. Finally, the red rectangle marks the component for storing the initial velocity of a braking process, which is needed to calculate the braking distance further down the line. This is achieved by using a block to detect a decrease of v_{max} , i.e. the block outputs a signal of 1 or 0, accordingly.

$$v_0 = v_{real} \cdot b + v_{0,old} \cdot \bar{b} \quad (3.3)$$

Equation 3.3 describes the logic. The initial velocity of a braking process v_0 equals the train's velocity multiplied with the result of the decrease detection block, denoted as b , added with the previously stored initial velocity $v_{0,old}$ multiplied with the inverse of b , denoted as \bar{b} . If there is a decrease of v_{max} , i.e. the train starts braking, we get $v_0 = v_{real} \cdot 1 + v_{0,old} \cdot 0 = v_{real}$. In any other case, we get $v_0 = v_{real} \cdot 0 + v_{0,old} \cdot 1 = v_{0,old}$, i.e. the value of v_0 does not change as no new braking process has begun.

This completes the description of this subsystem. It is worthy to note that, reaching back to chapter 2, this is the practical implementation of the brake valve \mathcal{M}_{bv} .

Figure 3.4 shows the subsystem responsible for the calculation of the traction force, the train's acceleration, the train's velocity and the train's traveled distance. The applied traction force scales with v_{dif} in a similar fashion to how p_{bp} does. Again, a one-dimensional lookup table is used to calculate the appropriate value for F_t .

$$H(n) = \begin{cases} 0.05 & \text{if } n = 1 \\ 0.45 & \text{if } n = 15 \\ 0.6 & \text{if } n = 20 \\ .. & \end{cases} \quad (3.4)$$

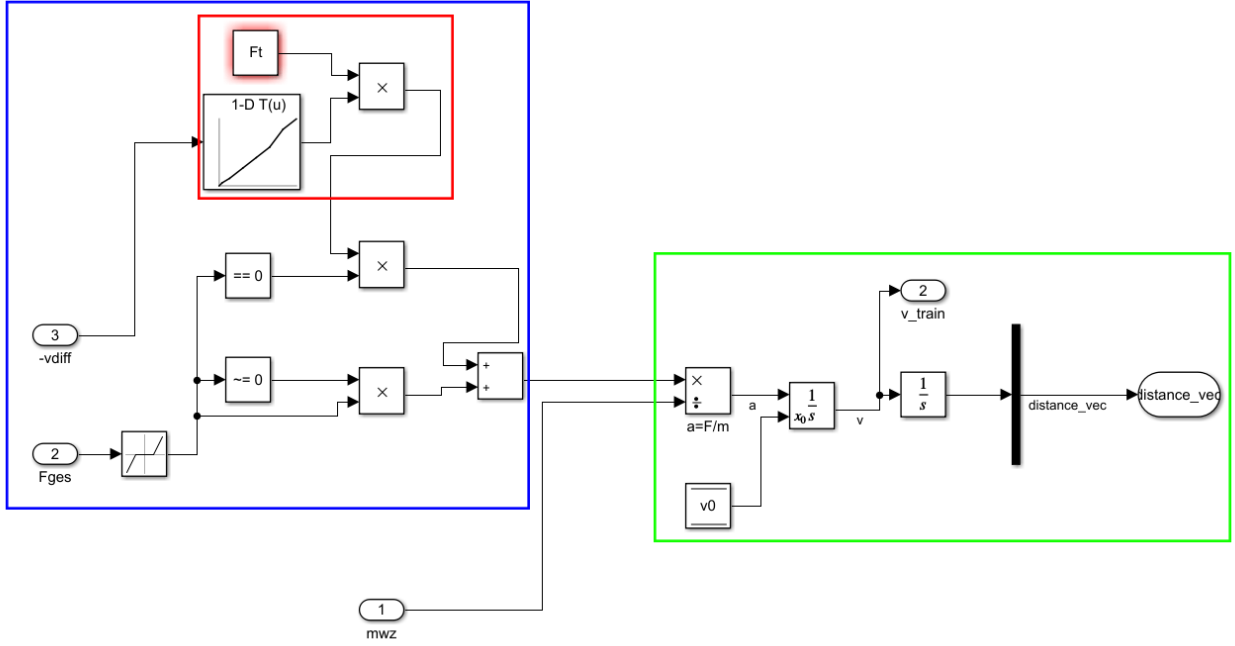


Figure 3.4: Expanded Model - Traction Force Calculation

Equation 3.4 shows an example for how the function for the table might look like. The key difference to the calculation of the braking pressure is that the output values of the function are coefficients which are then multiplied with the maximum possible traction force, rather than raw values like in equation 3.1. This part of the system is marked by the red rectangle.

Remember that we have established the train is at any given time either accelerating or decelerating. Since there is, as previously discussed, already a system in place to determine whether the train should engage its brakes, it now suffices to make the following examination: If the train is generating a braking force, i.e. F_b is greater than zero, it will apply no traction force, and vice versa.

$$F_t(n, t) = (H(n) \cdot F_{t,max}) \cdot (F_b(n, t) == 0) \quad (3.5)$$

Equation 3.5 describes the logic. $F_b(t) == 0$ is a relational operation, the result being either 1 if there is a braking force at a particular point in time t , otherwise 0. We can apply the same logic to the braking force F_b :

$$F_b(n, t) = F_B(n, t) \cdot (F_b(n, t) \neq 0) \quad (3.6)$$

With equations 3.5 and 3.6 we can now determine the actual force applied at any given point in time t :

$$F(n, t) = F_t(n, t) + F_b(n, t) \quad (3.7)$$

Since either of the summands is always zero, equation 3.7 outputs either the traction force or the braking force being applied at a point in time t , for a value of $v_{dif} = n$. This part of the

system is marked by the blue rectangle, and together with the brake valve completes the system responsible for controlling the train.

Let's reach back to our model \mathcal{M} from chapter 2, in particular to three elements of its set of model variables \mathcal{V} : The train's acceleration a , its velocity v , and its braking distance d . Since we can now determine the applied force $F(n, t)$, it is now also possible to calculate the values of these three variables.

$$F = m \cdot a \quad (3.8)$$

Once again making use of Newton's second law of motion, we may calculate the value of a by substituting $F(n, t)$ for F , and the combined mass of all wagons m_{total} for m :

$$a = \frac{F(n, t)}{m_{total}} \quad (3.9)$$

The train's velocity may now be calculated by integration of its acceleration a . In Simulink, this is achieved by feeding the signal value of a into a continuous-time integration block:

$$v = \int a \, da \quad (3.10)$$

Finally, the train's braking distance can be calculated by integration of its velocity v . This is once again achieved by usage of a continuous-time integration block:

$$d = \int v \, dv \quad (3.11)$$

This part of the system is marked by the green rectangle. This also completes the description of this subsystem. Looking back at our model \mathcal{M} , the last elements of \mathcal{V} left to deal with are the brake pipe \mathcal{M}_{bp} , the brake cylinder \mathcal{M}_{bc} , the wagon mass m , the train composition $comp$, and finally the braking force F_b .

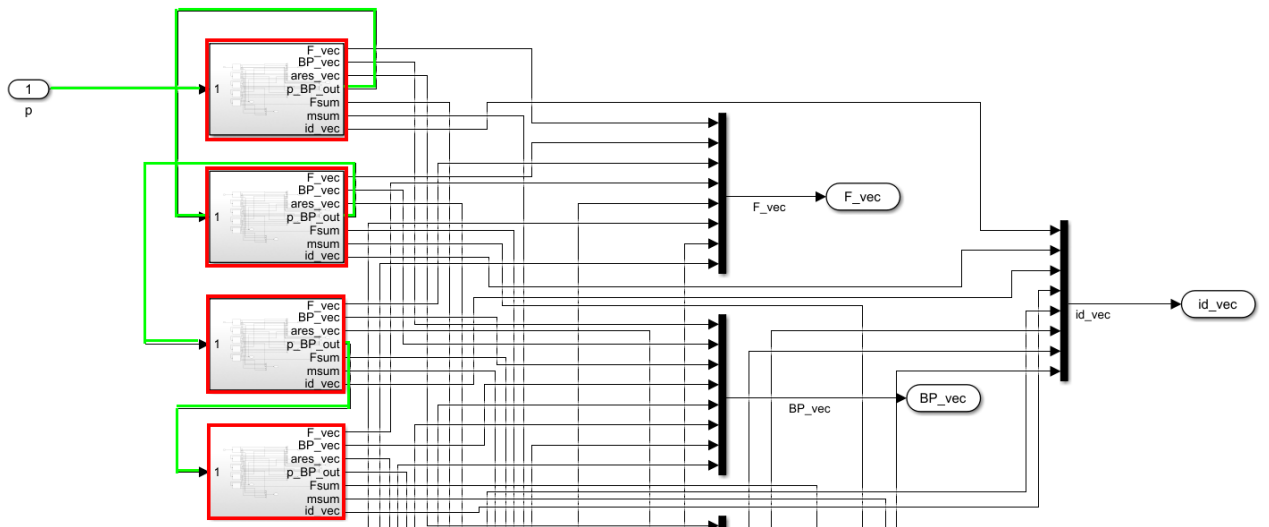


Figure 3.5: Expanded Model - Brake Pipe

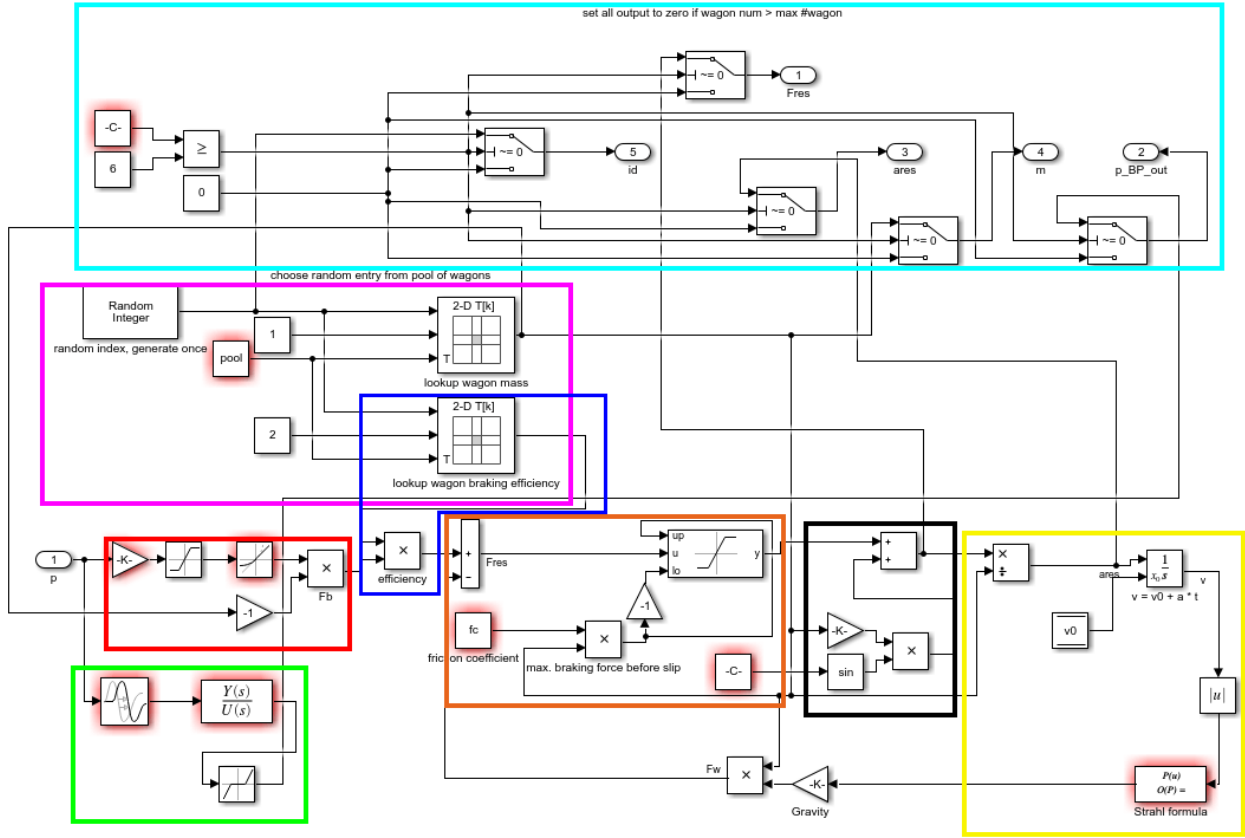


Figure 3.6: Expanded Model - Wagon

Figure 3.5 shows how the wagons (marked by red rectangles) are interconnected by the brake pipe (marked by green arrows). This is analogous to how the wagons were connected in the initial model from section 3.2.

Figure 3.6 shows the expanded model of the freight wagon. There are components which have remained largely unchanged, while others have been added or altered to account for the changes required to implement \mathcal{M} . To recapitulate, the initial model's freight wagon comprised of three components: The brake cylinder \mathcal{M}_{bc} , the brake pipe \mathcal{M}_{bp} , and the driving resistance. The implementation of the brake pipe has remained largely unchanged, the only exception being the addition of a dead zone to filter out very small values of p_{bp} , ranging from -0.005 to 0.005 . This was necessary as these small values interfered with the simulation result in a negative way. The implementation of the brake pipe \mathcal{M}_{bp} is marked by the green rectangle.

The implementation of the brake cylinder \mathcal{M}_{bc} has also remained largely unchanged from the initial model, though the wagon's mass used to calculate $F_{b,max}$ is now read dynamically rather than being a static value (more on this later). The brake cylinder is marked by the red rectangle. The implementation of the calculation of the train's driving resistance has too remained unchanged except for the wagon's mass, used to determine the acceleration, being read dynamically, analogous to the brake cylinder. This component is marked by the yellow rectangle.

According to the definition of \mathcal{M} , the following factors still need to be accounted for: The wagon mass m , the train composition $comp$, the track gradient α , and the wheel/rail friction coefficient μ . It is therefore necessary to add new components implementing these to the wagon model.

While the initial model's wagons did indeed have a mass, it had a static value, and it was the same for all 40 wagons, making them identical and indistinguishable from one another. This is of course undesirable as it poorly reflects reality. To make up for this flaw, a pool of 500 wagons has been randomly generated via a python script:

```
import csv
import random

with open('pool.csv', 'w', newline='') as pool:
    fieldnames = ['Wagon_ID', 'Mass', 'Braking_eff']
    writer = csv.DictWriter(pool, fieldnames=fieldnames)

    writer.writeheader()
    for i in range(0,500):
        mass = random.randrange(12000, 90000, 100)
        braking_eff = round(random.uniform(0.75, 0.95),2)
        writer.writerow({'Wagon_ID': i, 'Mass': mass, 'Braking_eff': braking_eff
                          })
```

This creates a table with 500 entries of the following structure:

Wagon_ID	Wagon Mass	Braking Efficiency
0	53800	0.83
1	60800	0.77
...
499	66600	0.81

Table 3.1: Wagon pool structure

Each wagon contained in the pool is identifiable by its ID and has the properties wagon mass and braking efficiency, i.e. a characteristic for the integrity of its brakes. Both properties are randomly generated for each wagon, the mass being assigned a value between 12000 and 90000 kilograms (with a step size of 100 kilograms), and the braking efficiency following a uniform distribution between 75 and 95 percent integrity. In order to read the values from the table, during simulation, each of the 40 wagon subsystems generates a random index between 0 and 499, i.e. each wagon is assigned an ID. By utilizing two direct lookup table blocks, the row with

the assigned ID is selected, and the corresponding values for mass and braking efficiency are read from the table. The implementation of this component is marked by the pink rectangle. The value of m is then used to calculate F_b , a , the driving resistance, the maximum braking force before slip, and the overall mass of the train m_{total} . The value of the braking efficiency is multiplied with F_b , meaning a lower integrity will result in a lower effective braking force. This component is marked by the blue polygon.

The next element of \mathcal{V} to address is the train composition *comp*. In the initial model, *comp* is static, meaning the train is always comprised of 40 identical wagons. There is now a system in place that allows for wagons with diverse properties; we now need to design a system that allows for a variable number of wagons. In the Simulink model, the train is made up of 40 subsystems, each representing one freight wagon. The obvious approach would be only having as many of these systems as desired, e.g. for a train consisting of 27 wagons, there would only be 27 wagon subsystems in the model. Since Simulink however offers no straightforward way of adding or removing subsystems during simulation, a workaround is necessary. The solution taken here is using switch blocks to set output to zero for all unwanted subsystems, effectively removing their impact on the simulation. Each of the 40 models is assigned a constant representing their position; for each simulation, the desired number of wagons the train should encompass is specified as simulation input. During simulation, each subsystem compares this value to its position index. If their index is greater than the desired number of wagons, the switch blocks are set to output zero. For example, take a train of 15 wagons. The subsystems with position indices one to 15 will produce output, and the subsystems with indices 16 to 40 will not. This component is marked by the cyan rectangle.

The track gradient α is used to calculate the additional braking or traction force resulting from the track's pitch angle.

$$F_N = G_{F,Z} \cdot f_N \quad (3.12)$$

Equation 3.12 is used to calculate the inclination force F_N , where $G_{F,Z} = m \cdot 9.81$ is the weight force acting on the wagon, and $f_N = \sin(\alpha)$ is the inclination force number ([Wen03], p.95). Accordingly, this force is calculated during simulation and then added to (or subtracted from) the generated braking force F_b to determine the effective braking force, taking into account the slope of the train track. This component is marked by the black rectangle.

As mentioned in chapter 2, the wheel/rail friction coefficient μ determines how much braking force may be applied before sliding occurs, negatively impacting the braking performance. Since the exact mechanics of how sliding affects the braking process go beyond the scope of this work, a different approach has been chosen to incorporate μ .

$$F_{b,e} = F_{b,max} \cdot \mu \quad (3.13)$$

Equation 3.13 is used to calculate the braking force before sliding would occur, $F_{b,e}$ by multiplying the maximum possible braking force $F_{b,max}$ with the wheel/rail friction coefficient μ . The result is then used to limit the applied braking force F_b so that sliding is avoided, utilizing a saturation dynamic block. This still satisfies $R_{\mu,F_b} \in \mathcal{R}$. This component is marked by the brown rectangle.

This completes the description of the Simulink model. Reaching back to chapter 2, all elements of \mathcal{V} , \mathcal{R} and \mathcal{C} have been accounted for, and we can therefore conclude our theoretical model \mathcal{M} has been implemented. It may now be used for simulation and data generation.

3.4 Outlook

While the model defined in chapter 2 satisfies the requirements for this work, there are many more factors that might be incorporated in order to achieve a more accurate representation of reality. Some of these factors shall be discussed here briefly. So far, the generated braking force F_b is calculated by transforming the braking pressure p_{bp} into a coefficient, and applying that coefficient to the maximum braking capacity $F_{b,max}$.

$$F_b = \xi \cdot \sum P_N \quad (3.14)$$

Equation 3.14 offers a more detailed way of calculating F_b , where $\sum P_N$ is the sum of all applying forces actuating the brake shoes on the wheel tread, or the brake pads against the discs, respectively; the proportionality coefficient ξ might be the friction coefficient between the brake shoes and the wheel, or

$$\xi = \frac{4 \cdot r_m}{D_o} \cdot \mu_d \quad (3.15)$$

for disc brakes, where r_m is the medium friction radius, D_o is the wheel diameter, and μ_d is the friction coefficient between brake pad and disk ([Cru12], p.51).

As the focus of this work lies on the braking process, the modeling of traction force is kept rather simple. It might however be desirable to create a more sophisticated model, in order to achieve a more realistic result. For example, as of right now, the applied traction force depends only on the discrepancy between current and maximum velocity; in reality, the maximum traction force is also limited by wheel/rail adhesion just as the braking force is ([Wen03], p.157). One might also want to factor in differences between various engines.

Chapter 4

Data Generation

Introduction The implemented model may now be utilized to simulate braking processes, generating output data describing the behavior of the train during the process.

4.1 Simulation

To recapitulate: We have defined a theoretical model for a train's braking process in chapter 2, and implemented this model in Simulink in chapter 3. The aim of this work, however, is not only building a model, but also generating a data set describing the train's behavior during its phases of breaking. We can utilize Simulink to run a large number of simulations with varying combinations of input parameters. Each simulation represents one train ride, each ride consisting of multiple phases of accelerating, braking and keeping velocity.

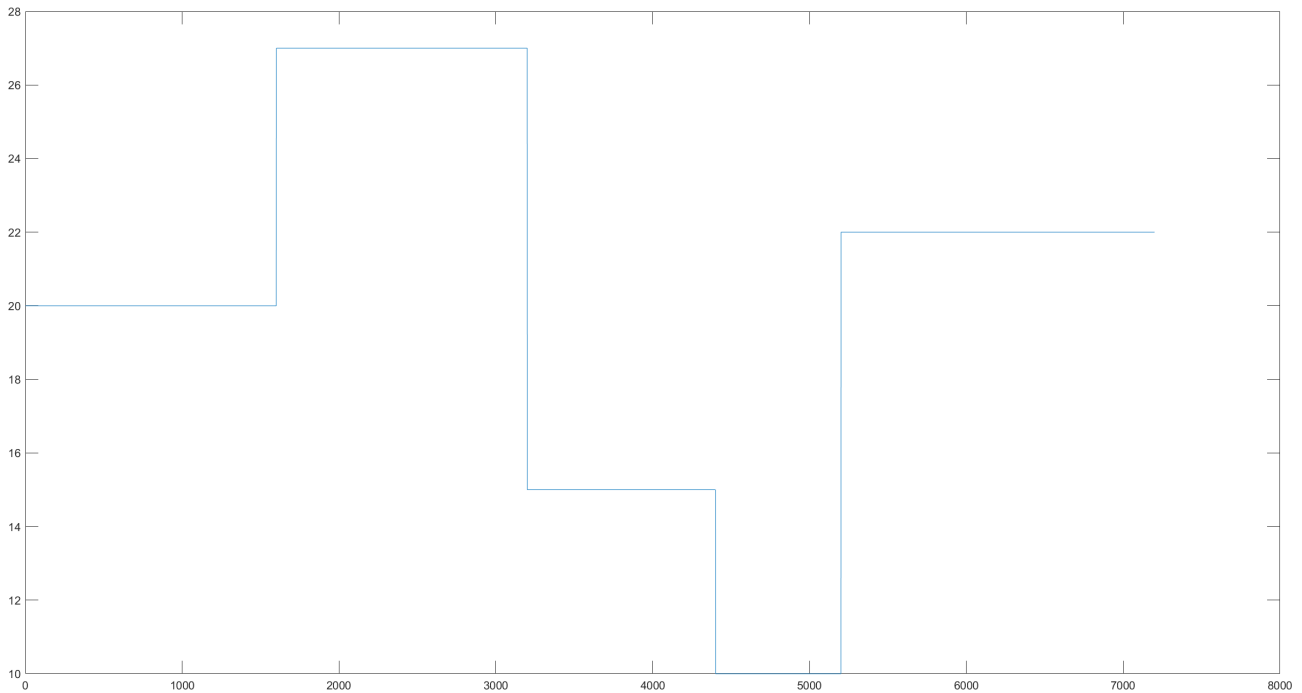


Figure 4.1: Simulation Input

Figure 4.1 visualizes how a track profile might look like. It stipulates the maximum allowed

velocity for discrete points in time, as well as the overall duration of the simulation; the train either brakes or accelerates accordingly in order to match the corresponding value. An example: Let the simulation time be 1200 seconds, and $v_{max}(1200) = 20 \frac{m}{s}$. Let the train's velocity v be $25 \frac{m}{s}$. The train will engage its brakes (or continue braking, if they have already been engaged) in order to bring v down to $20 \frac{m}{s}$. If v were $15 \frac{m}{s}$, the train would accelerate instead. The track profile is one of many input parameters, all of which determine the course of the simulation.

4.2 Matlab Code

A Matlab script is used to configure the simulation input, execute the simulation, and write the simulation output into a tabulator-separated file. At the start of execution, the workspace (where variables are stored) should be cleared. Also, the number of cores to use for parallel processing must be set (more on this later):

```
clear all
clc
warning ('off','all');
numcores = 10;
```

Generally speaking, simulation input can be divided into two categories: static and dynamic parameters. They, for the most part, correspond to the constants and variables defined in chapter 2. The static parameters, i.e. constants, remain the same for every single simulation run, while the dynamic parameters, i.e. variables, may be different for any two simulation runs.

```
RBD = 5;
VBD = 3.5;
l = 18;
c = 250;
```

These constants all correspond to the brake pipe \mathcal{M}_{bp} , where RBD is the regular operations pressure (i.e. the pressure on the brake pipe when brakes are disengaged), VBD is the full breaking pressure (i.e. the pressure on the brake pipe when brakes are fully engaged), l is the physical length of the brake pipe l_{bp} (although l_{bp} is defined as a variable in \mathcal{M}_{bp} , it is assumed all wagons have the same physical length for the sake of simplicity), and c is the propagation velocity of the pipe's medium v_{bp} .

```
tf = 4;
tl = .1;
```

<TODO> </>

```
p0 = 0;
Pres = 0/1000*[5.7/771 0 1.6];
alpha = 0.9;
BPnum = [0.3 alpha];
BPden = [1 alpha];
```

<TODO> </>

All static parameters have now been initialized. They, along with the track profile, will remain unchanged for all simulations; what sets any two simulations apart is their variable input parameters.

```
tmax = 3600;
nmax = tmax * 2;
t = linspace(0, tmax, nmax);
u = [10*ones(1000*2,1); 15*ones(500*2,1); 25*ones(1200*2,1); 12*ones(600*2,1);
      0*ones(300*2,1)];

simin.time = t;
simin.signals.values = u;

trackgradient = 0;
```

The code above shows the definition of the track profile. `tmax` specifies the duration of the simulation in seconds, so in this case one hour. `nmax` is the number of discrete data points; with `nmax = tmax * 2`, there are on average two data points for every one second of simulation time. `t` is a linearly spaced vector, i.e. a vector of `nmax` evenly spaced points between 0 and `tmax`, which is then set as time input using `simin.time = t`; `u` is the vector which holds the data points, with the number of data points being equal to the value of `nmax`. The data points stipulate the maximum allowed velocity v_{max} for every discrete point in time. For example, `10*ones(1000*2,1)`; means that for the first 2000 points in time, or for the first 1000 seconds of simulation time, the value of v_{max} is $10 \frac{m}{s}$. Continuing with the above example, v_{max} then increases to $15 \frac{m}{s}$ for the next 1000 points in time, or the next 500 seconds of simulation time. This vector `u` is then set as input signal using `simin.signals.values = u`; Finally, `trackgradient = 0`; specifies the average gradient of the track (in this case, the track neither goes uphill, nor downhill).

The goal is to create large quantities of data, i.e. at least hundreds of gigabytes. Since one single simulation run takes, on average, five seconds to complete, and one simulation run produces approximately one megabyte of output data, producing 500 gigabytes of data would take 2,500,000 seconds, or almost 29 days. Obviously, this is not acceptable. Fortunately, Simulink simulations may be run in parallel, where hardware is the limiting factor rather than time. Albeit, simulations are very expensive, and one will quickly run out of memory if too many simulations are run simultaneously (Using 20 cores, i.e. running 20 simulation in parallel, lead to the server freezing up). Using too many cores also lead to the Matlab engine timing out, though whether this was due to memory restrictions or limitations of the engine is unclear. Anyway, using ten CPU cores for parallel processing proved to be stable.

```
wagons = 1:1:40;
friction = 0.05:0.01:0.78;
tracforce = 200000:1000:400000;
```

The remaining elements of \mathcal{V} left to address are wagon mass m , train composition *comp*, and wheel/rail friction coefficient μ ; furthermore, the maximum traction force the locomotive can

generate. The wagon mass is handled by the wagon pool (section 3.3) and thus is not part of the Matlab script. All possible combinations of the remaining three variables are created using nested loops. Their values range from one to forty for the number of wagons (*comp*), step size one, 0.05 to 0.78 for wheel/rail friction coefficient (μ), step size 0.01, and 200,000 to 400,000 for traction force, step size 1,000. Therefore, the total number of possible combinations is $40 \cdot 74 \cdot 200 = 592000$, i.e. for every distinct track profile, there are 592,000 simulation runs.

```
track = 1;
allruns = 1;
for i = length(tracforce):-1:1
    for j = length(friction):-1:1
        for k = length(wagons):-1:1
            in(track) = Simulink.SimulationInput('Simulation_v2');
            in(track) = in(track).setVariable('num_wagons',wagons(k));
            in(track) = in(track).setVariable('fc',friction(j));
            in(track) = in(track).setVariable('Ft',tracforce(i));

            in(track) = in(track).setVariable('alpha',alpha);
            in(track) = in(track).setVariable('BPden',BPden);
            ...
            in(track) = in(track).setVariable('VBD',VBD);

            Ft(track) = tracforce(i);
            fc(track) = friction(j);

            track = track + 1;
        end
    end
end
```

An array to hold all simulation input is created using `in(track) = Simulink.SimulationInput('Simulation_v2');` in the innermost for-loop. Each array field holds the input data for one simulation run. These parameters are set using `in(track) = in(track).setVariable(var);`, where `var` are all the constants and variables previously defined. `Ft(track)` and `fc(track)` hold the value for traction force and wheel/rail friction coefficient for each simulation, since these values can not be found in the simulation output.

```
parpool(numcores);
out = parsim(in,'ShowProgress','on');
```

Creating one large array for all 592,000 simulations leads to the Matlab engine freezing, presumably due to running out of memory, so a smaller batch size is used. After the two inner for-loops have elapsed, i.e. array size is $74 \cdot 40 = 2960$ entries, a pool of ten parallel workers is created, and the input array is passed to the Matlab engine for simulation, using `out = parsim(in);`. The simulation output is stored in `out`, which is also an array with 2960 entries.

```
matrix = [];
parfor l = 1:length(out)
```

```

    tmp = Write(1 + allruns, out(1).get('velocity'), out(1).get('force'), out(1)
                .get('pressure'), out(1).get('distance'), out(1).get('acceleration_neg'), out(1).get('ids'), u, trackgradient, Ft(1), fc(1));

    matrix = [matrix; tmp];
end
allruns = allruns + length(out);
writematrix(matrix, 'output/output2.tsv', 'FileType', 'text', 'WriteMode', 'append', 'Delimiter', 'tab');

track = 1;
delete(gcf('nocreate'));
end

```

Writing of the output is also done in smaller batches. The best approach to minimize the number of file accessions is of course writing everything into one big matrix (592,000 times 300 output lines per simulation would result in a 177,600,000 line matrix!), thus storing in memory, and then writing the matrix in one go. Unfortunately, growing an array by assignment or concatenation can be expensive. For large arrays, Matlab must allocate a new block of memory and copy the older array contents to the new array as it makes each assignment. The other extreme would be writing line by line, but the resulting large number of file accessions required makes this just as unfeasible (again, that would be approximately 177,600,000 file accessions!). Therefore, finding a middle ground is necessary, and a batch size of 2960, i.e. the size of one simulation batch, works relatively well, although this could surely be optimized with a bit of runtime analysis. To optimize resource usage, writing is also done in parallel. This is possible because each simulation is identifiable by a unique ID, and thus synchronicity is not required, i.e. the simulations need not be written in order. An empty matrix is created using `matrix = [];`. The `Write()` function compresses all different output parameters from one simulation into one temporary matrix `tmp` and appends this temporary matrix to the output `matrix`. After all 2960 simulation outputs have been stored into `matrix`, it gets written to an output tabulator-separated file, the track counter is reset to one, and the parallel pool gets deleted so that the next batch of simulations may be run.

```

function ret = Write(id, velocity, force, pressure, distance, acceleration,
                    wagon_ids, u, grad, ft, fc)

    numrows = get(force, 'Length');
    wagon_ids = double(wagon_ids);
    time = force.Time;
    matrix = [ ];

    for i = 1:1:numrows
        f = getdatasamples(force, i);
        p = getdatasamples(pressure, i);
        v = getdatasamples(velocity, i);
        a = getdatasamples(acceleration, i);
        d = getdatasamples(distance, i);

        row = [id, time(i), f, p, v, a, d, wagon_ids, grad, ft, fc];
        matrix = [matrix; row];
    end

```

```

    end

    ret = matrix;
end

```

The `Write()` function takes all output parameters of a single simulation for its arguments. These are the unique ID of the simulation, a time-series object storing the train's velocity over the course of the simulation, a time-series object storing the wagons' applied braking force over the course of the simulation, a time-series object storing the wagons' braking pressure over the course of the simulation, a time-series object storing the train's traveled distance over the course of the simulation, a time-series object storing the wagons' acceleration over the course of the simulation, a vector storing the IDs of the wagons making up the train, a vector storing the track profile, the track gradient, the train's traction force, and the wheel/rail friction coefficient. The time-series objects all have the following general structure:

Timestamp	Column 1	Column 2	Column n
0	a	b	c
...
3600	x	y	z

Table 4.1: General structure of a time-series object

For one simulation, all time-series objects have the line count, as well as the same timestamps, so it suffices to store the corresponding properties of one time-series object, using `numrows = get(force, 'Length');` and `time = force.Time;`, respectively. An empty matrix is then created using `matrix = []`; which will store all output parameters. A for-loop is used to iterate over the time-series objects, creating a vector for each row that contains the simulation ID, the timestamp for the current row, the column values of all time-series objects for the current row, the wagon IDs, the track gradient, the traction force, and the wheel/rail friction coefficient. This vector is then appended to the overall matrix. After the for-loop has elapsed, the `matrix` object gets returned.

4.3 Data Structure

simID	Timestamp	$F_{b,0}$	$F_{b,1}$	$F_{b,n}$	$F_{b,39}$
0	0	0	0	0	0
0	t	$f_{t,0}$	$f_{t,1}$	$f_{t,n}$	$f_{t,39}$
0	3600	$f_{3600,0}$	$f_{3600,1}$	$f_{3600,n}$	$f_{3600,39}$
1	0	0	0	0	0
\vdots					
n	3600	$f_{3600,0}$	$f_{3600,1}$	$f_{3600,n}$	$f_{3600,39}$

Table 4.2: Structure of the output table

$P_{bp,n}$	v	a_n	d	WagonIDs	α	F_t	μ
0	0	0	0	55 327	0	200000	0.45
\vdots	\vdots	\vdots	\vdots	55 327	0	200000	0.45
$p_{3600,n}$	v_{3600}	$a_{3600,n}$	d_{3600}	55 327	0	200000	0.45
0	0	0	0	476 202 194	0	320000	0.37
\vdots	\vdots	\vdots	\vdots	476 202 194	0	320000	0.37
$p_{3600,n}$	v_{3600}	$a_{3600,n}$	d_{3600}	476 202 194	0	320000	0.37

Table 4.3: Structure of the output table, continued

Tables 4.2 and 4.3 show the data structure generated by the `Write()` function. For each simulation, Simulink creates one time-series object for every monitored parameter. In particular, these are the braking force F_b , the braking pressure P_{bp} , the velocity v , the acceleration a , and

the traveled distance d . These time-series objects all have the same general structure shown in table 4.1, where the number of columns corresponds to the amount of monitoring points; F_b , P_{bp} and a , which are monitored for each wagon, have 41 columns (40 wagons plus timestamp), and v and d , which are monitored for the entire train, have two (one train plus timestamp). As mentioned earlier, for one simulation, all time-series objects have the same number of rows. It is therefore possible to put all time-series objects into one wide table. We can then also add the input parameters to this table. In particular, these are the ids of the wagons used in the simulation (*comp*), the track gradient α , the traction force F_t , and the wheel/rail friction coefficient μ . In theory, these values only need to be written once, but the subsequent rows should be filled regardless to avoid empty cells, see table 4.3 for reference.

One is now presented with two choices: Either write a new table file for every simulation (i.e. have hundreds of thousands of small files), or write everything into one single large table file. As we will generate large quantities of data, creating only one table is the preferred choice since it is more suitable for big data processing. In fact, as of now, two of these tables have already been loaded into a hadoop distributed file system, making further analyses possible. Subsequently, a column containing the simulation identifier needs to be added to the table structure to make differentiation between simulations possible. This leads to the final table structure shown in tables 4.2 and 4.3. Remember that there are 592,000 simulations for each track profile, therefore there must be 592,000 unique ids, each referencing the results of one simulation. Each simulation, in turn, has approximately 300 rows, the exact count determined by Simulink. The time-series objects have 41 columns for wagon output, and two columns for train output, respectively; all together, there are $3 \cdot 41 + 2 \cdot 2 + 6 = 133$ columns in one such file. It is worthy to note that even though there may be less than 40 wagons in one simulation, the wagon-specific time-series objects (F_b , P_{bp} and a) will always have 41 columns, with the unused columns filled with zeros (Remember that in chapter 2, it was stated output for unwanted wagons is set to zero values).

One parameter, however, is still missing from the table: The track profile. It is a vector of the following structure:

#	v_{max}
0	10
1	10
\vdots	
2200	15
\vdots	
7199	0

Table 4.4: Track profile structure

The first column (#) is the index for the data points, and the second column is the corresponding value for maximum velocity v_{max} at this particular point in time. A few choices present themselves of how to add to the output table. It could be added to every simulation; this would lead to massive overhead, however, since ~ 6900 rows would need to be filled for every single simulation. Furthermore, it is unnecessary to add the track profile to every simulation, since it does not change, compared to varying input parameters like the number of wagons. Therefore, one might either write the vector once, to the end of the table, or write it to a separate file. Appending it to the end of the table still leads to slight overhead in terms of disk space, since for each simulation, two additional columns would need to be filled with zeros; also, this would lead to the last 7199 rows of the output table having different semantics than the previous entries, which would need to be highlighted somehow. The easiest approach, which has been chosen accordingly, is therefore writing the track profile to a separate file, which can then be linked to the output table further down the line.

4.4 Analysis of generated Data

<TODO> </>

4.5 Performance

Finally, a few words about performance aspects. Simulink, while being intuitive and rather easy to learn, might be lacking in terms of raw performance. As mentioned earlier, single simulations can be computed in a matter of seconds; when it comes to larger quantities however, as in our case, these few seconds quickly start to add up. As a reference, the average time it took to complete all 592,000 simulations for one track profile was around four days, or 96 hours! It must be noted that Simulink does offer some ways to improve performance. For one, a model

inspector can be used to check for faulty logic in the underlying model, which might hamper performance (e.g. unnecessary loops, ill-configured blocks etc). There are also acceleration and hyper-acceleration compilation options which increase performance. The acceleration mode has, in fact, been used for this work, and it has sped up the process considerably. The hyper-acceleration mode, however, does not allow for model output and is therefore utterly unsuited for this work since generating data becomes impossible. Fortunately, the Matlab engine offers parallel processing, so the performance may yet be considerably improved if one possesses enough computing power. As mentioned earlier, the Matlab engine is quite resource-hungry, however.

Since performance was no major factor in this work, no real tangible analyses have been made (e.g. runtime measurements, hardware utilization etc.). Surely, in further work, the performance might be optimized; the current state is satisfactory anyhow. There are a few alternatives to Matlab and Simulink on the market, like Modelica or Scilab's Xcos, but whether these pieces of software offer a better performance necessitates further research. This could be the focus of future works continuing this topic.

Chapter 5

Conclusion

The aim of this work has been the creation of an artificial data set describing the braking processes of cargo trains in order to compensate for the lack of actual data from real life operations. To achieve this, the following working steps needed to be taken: First, it was necessary to develop an understanding of train operations in general, and the braking process of a train in particular. Here, the focus primarily lay on identifying the key parameters influencing the process. Once these had been worked out, it was possible to define a theoretical model \mathcal{M} for the braking process. The model allowed us to formalize the inner workings of the system and how its different parts relate to each other, therefore serving as a basis for a practical implementation.

Since a theoretical model can not be used for simulation and data generation purposes, the subsequent working step was the implementation of the model \mathcal{M} . Groundwork for this had already been laid in form of a rudimentary model of a cargo train of fixed length, breaking from an initial velocity to a full halt. As this model had been created with Mathwork's Simulink, it was decided to use this piece of software for this work, too. For the modeling work, Simulink proved to be an excellent choice, as it is intuitive to use and did not require a disproportionate amount of work to get used to. The expansion of the initial model, necessary to transform it into a complete implementation of the previously defined theoretical model, required quite a bit more time and effort, however, especially for someone not familiar with the field of railway vehicle engineering. Apart from the obvious challenges, a few problems appeared from time to time which were quite tricky to fix, mainly because Simulink does not offer much in terms of debugging.

The implemented Simulink model could now finally be used to simulate a variety of braking processes, and record the behavior of the trains during these processes to create the artificial data set. In theory, this should have been the least laborious working step; considering the data set should be suitable for big data processing, finding a *stable* way of generating large quantities of data proved to be a bit challenging. In the end, however, a satisfying result has been achieved, and more data sets may be generated, within an acceptable amount of time, as needed.

Looking onward, this work could serve as basis for future efforts. On the one hand, the data sets may be used for data analyses, examples of application being predictive maintenance or further automation of train operations, leading towards autonomous trains and optimized utilization of infrastructure. On the other hand, the model of the braking process may be

expanded further in order to achieve a more accurate representation of the real life process, and thus improve the quality of the generated data. Also, it might be feasible to optimize the performance of the data generation by means of runtime and code analysis.

Bibliography

- [BGS06] N. Bosso, A. Gugliotta, and A. Somà. Design and simulation of railway vehicles braking operation using a scaled roller-rig. WIT Transactions on The Built Environment, 88:869–883, 2006. DOI 10.2495/CR060851.
- [Cru12] Catalin Cruceanu. Train braking. In Xavier Perpinya, editor, Reliability and Safety in Railway, pages 30–74. Intech Open, 2012. DOI 10.5772/2660.
- [EP17] Manfred Enning and Raphael Pfaff. Güterwagen 4.0 - Der Güterwagen für das Internet der Dinge. Teil 1: Gesamtsystembetrachtung und grundlegendes Konzept. ETR — Eisenbahntechnische Rundschau, 66:12–16, 2017. ISSN 0013-2845.
- [Hav17] Volodymyr I. Havryliuk. An overview of the ETCS braking curves. 2017. DOI 10.15802/ecsrt2017/136811.
- [Pfa17a] Raphael Pfaff. Analysis of big data streams to obtain braking reliability information for train protection systems. In Asia Pacific Conference of the Prognostics and Health Management Society, Jeju, Korea, 2017.
- [Pfa17b] Raphael Pfaff. Modeling of braking curves: White, grey and black box approaches. In Transrail Connection, 2017. DOI 10.13140/RG.2.2.15146.72640.
- [Pfa19] Raphael Pfaff. Ihha wagon 4.0, 2019.
- [Wen03] Dietrich Wende. Fahrdynamik des Schienenverkehrs. Vieweg+Teubner, Wiesbaden, 2003. DOI 10.1007/978-3-322-82961-0.

List of Figures

1.1	Spacing paradigms [Pfa17b]	12
1.2	Braking curves [Pfa17b]	13
3.1	Initial Model	24
3.2	Initial Model - Wagon	25
3.3	Expanded Model - Pressure Calculation	26
3.4	Expanded Model - Traction Force Calculation	28
3.5	Expanded Model - Brake Pipe	29
3.6	Expanded Model - Wagon.....	30
4.1	Simulation Input	35
A.1	Initial Model - Subsystem.....	53
A.2	Initial Model - Simulation Input.....	54

List of Tables

3.1	Wagon pool structure	31
4.1	General structure of a time-series object	40
4.2	Structure of the output table	41
4.3	Structure of the output table, continued	41
4.4	Track profile structure	43

Appendix A

Data visualization

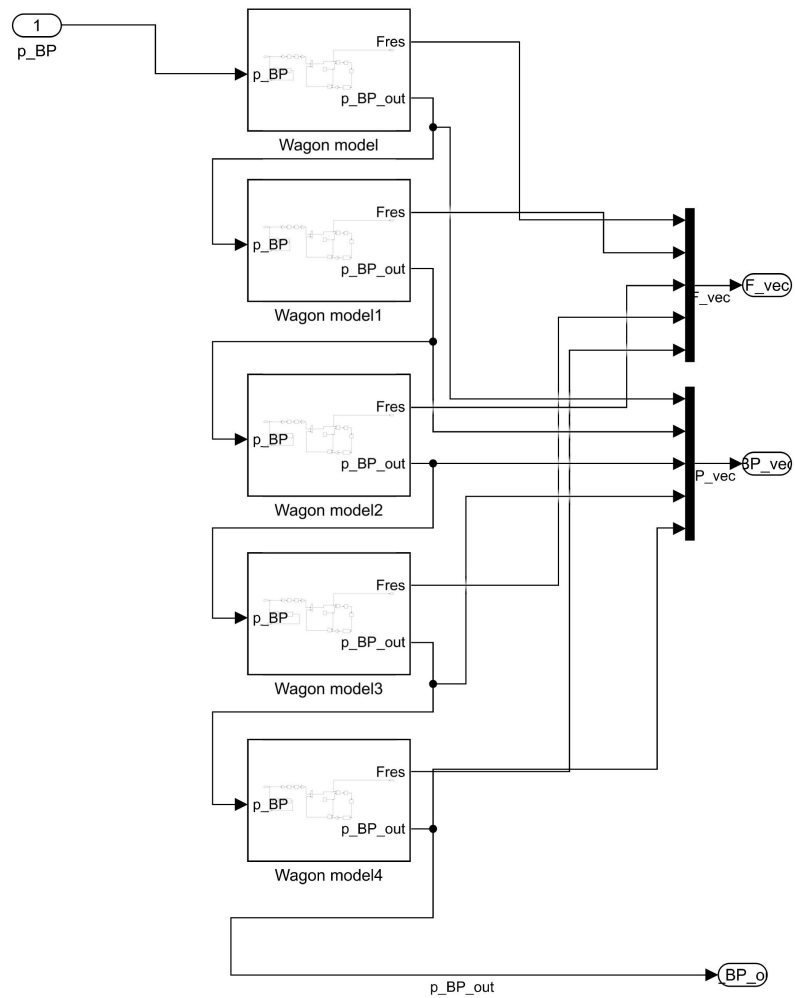


Figure A.1: Initial Model - Subsystem

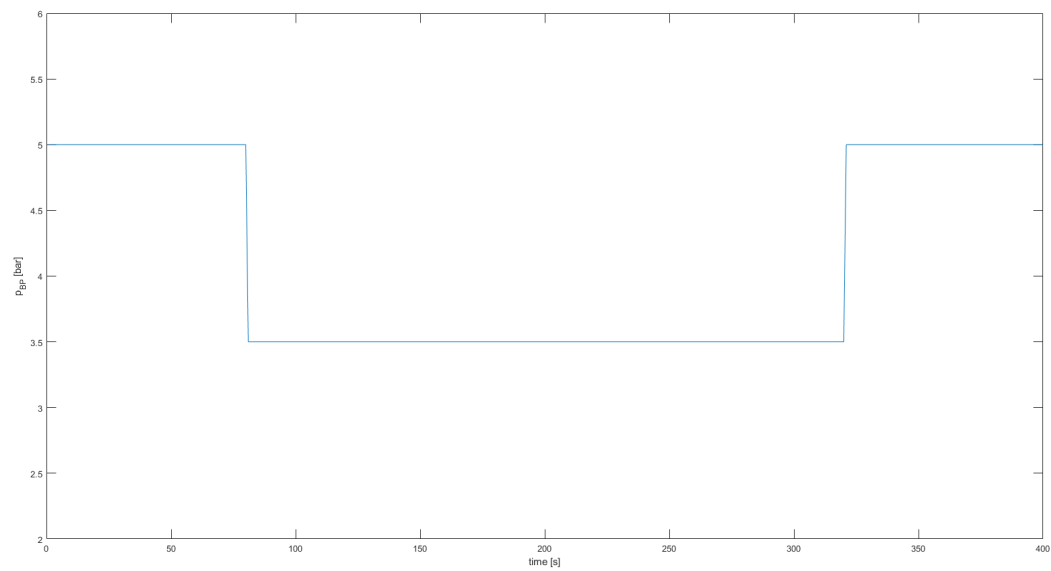


Figure A.2: Initial Model - Simulation Input