**Technische Universität Berlin**

**Institut für Mathematik**

# Numerical simulation of train traffic in large networks via time-optimal control

**C. Blendinger, V. Mehrmann, A. Steinbrecher and R. Unger**

**Technical Report 722-01**

**Preprint-Reihe des Instituts für Mathematik**
**Technische Universität Berlin**

**Report 722-01** **February 2002**

**Abstract**

We discuss the mathematical modelling of schedule based rail traffic. The model is used to develop efficient numerical simulation methods for the time optimal control of a large number of interacting trains in a large network. The time optimal control is used to model the realistic behaviour of driving in a network like that of Deutsche Bahn. It is used to allow a real time simulation with incomplete information on real train velocities. We present numerical examples that demonstrate the efficiency of the model and the simulation method.

**Keywords.** time optimal control, traffic simulation, mathematical modelling

Authors address:

Christoph Blendinger
TLC GmbH,
Weilburger Str. 26,
D-60326 Frankfurt, FRG
Christoph.Blendinger@tlc.de

Volker Mehrmann
Institut für Mathematik, MA 4–5,
Technische Universität Berlin
Str. des 17. Juni 136, D-10623 Berlin, FRG
mehrmann@math.tu-berlin.de

Andreas Steinbrecher
Institut für Mathematik, MA 4–5,
Technische Universität Berlin
Str. des 17. Juni 136, D-10623 Berlin, FRG
steinbrecher@math.tu-berlin.de

Roman Unger
Fakultät für Mathematik,
Technische Universität Chemnitz
D-09107 Chemnitz, FRG
roman.unger@mathematik.tu-chemnitz.de

This research was supported by *Bundesministerium für Forschung und Technologie* as Project 03-MEM4B1 of the program 'Neue Mathematische Verfahren in Industrie und Dienstleistungen',

# 1 Introduction and Preliminaries

In this paper we discuss the mathematical modelling of schedule-based rail traffic in large rail networks and present some examples of numerical calculations for this model of rail traffic.

Our motivation is the need for a simulation tool in real-time decision making for dispatching units in railway systems. Such dispatching units (e.g. the "Betriebszentralen" of German Railways/Deutsche Bahn) have to guide the planned trains according to their schedules. There are some technical devices to trace the current location of the trains. In case of detecting a deviation of some train from its schedule the dispatching unit is able to change the departure times, the tracks or routes and within some limits the maximum speed of trains. Evaluating the proposed dispatching actions needs a powerful simulation tool for human and even for machine based decision making.

Other duties of a dispatching unit (here not further mentiond) are the handling of train or line break-downs and the scheduling of extra trains.

In order to simulate the movement of all trains in a network we need a model that describes the following issues:

- the train movement, i.e., the dynamic equations;
- the constraints for the movement, like e.g. global velocity constraints or the maximum power of the engine;
- the detailed properties of the pathways of the trains (e.g. local velocity constraints, slopes);
- restrictions for departure or arrival times due to the schedule;
- the strategies that are used to control the dynamics of the train;
- interaction between the trains induced from the signal system;
- interaction between the trains induced from schedule based dependencies (e.g. connections for passengers, sequencing of trains).

It should be noted that neither a complete net model of the underlying infrastructure nor a complete model of the signal system is needed inside the simulation tool. The key idea of our model is that only the interactions between trains induced by the infrastructural properties (especially the signal system) are included in this model.

This allows a mathematical formulation of the problem which is prepared for the application of efficient (real time) numerical methods to solve the simulation problem.

First we present a standard model for one-train-dynamics without any interactions with other trains. In a second step we add the different kinds of interactions between trains in a unifying way as the proposed model of the movement of many trains in a network. For some prototypic (two-train-)situations the numerical simulation of their movements is shown.

# 2 One train: a control problem formulation

We use the simple model of the movement of one mass point to model the movement of a single train as a control problem. We denote the position of a fixed point of the train

(preferably the head of the train) at time $t$ by $s(t)$ and the velocity of this point by $v(t)$ and use the combined position/velocity vector

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

as state of the train at time $t$. For a given time grid $t_0, t_1, ...$ we have (initial) conditions describing the position and/or velocity at times $t_i$

$$\begin{bmatrix} s(t_i) \\ v(t_i) \end{bmatrix} = x(t_i) = \begin{bmatrix} s^i \\ v^i \end{bmatrix} \tag{1}$$

describing e. g. the beginning of a journey, as well as endpoints or stops.

In this way we can model the movement of every train as a classical linear control problem, where the control $u = u(t)$ is given by a force $F$ applied to the train. The equations of motion are then given by Newton's law

$$m\ddot{s}(t) = F(t) = u(t). \tag{2}$$

The total force acting on the train consists of several parts:

- A part $u_f(s, v)$ depending on position and velocity of the train, which cannot be influenced by the engineer (or an automatic driving control system). This includes air resistance and friction as well as gravitational effects driving down-hill or up-hill, see e.g. [6]. The parameters used in this model are the same as those used in the dynamical model of RUT-0, the timetable construction program used of Deutsche Bahn.
- A part $u_v(t)$ that can be influenced by the engineer and contains the acceleration or braking force which is constrained by the maximal braking force $u_{min}$ and the maximal power $P_{max}$ of the engine.

A simplified model ignoring some frictional effects at low velocities (see [6]) shows an inversely proportional dependency of the maximal acceleration force on the velocity: With the work $W$ we have the power as $P = \frac{\mathrm{d}W}{\mathrm{d}t}$ and with $W = \int F \, \mathrm{d}s$ we obtain $P = Fv$. With constant maximal power $P_{max}$ of the engine we obtain for nonzero velocities a maximal acceleration force of the form $u_{max}(v(t)) = P_{max}/v(t)$.

Combining these observations we obtain the constrained control

$$u = u_f(s, v) + u_v(t), \qquad u_v(t) \in [u_{min}, u_{max}(v(t))]. \tag{3}$$

With

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

we can describe the dynamics of the train as a linear control problem

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{4}$$

or in matrix notation

$$\begin{bmatrix} \dot{s}(t) \\ \dot{v}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} s(t) \\ v(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u(t), \tag{5}$$

subject to the constraints (3) as well as other constraints such as

- the maximal velocity $v_{max}(s)$ and the restriction that the train cannot drive backwards, i.e.,

$$v(s(t)) \in [0, v_{max}(s(t))] \quad \text{for } s \in [s^0, s^n], \tag{6}$$

- reaching of a certain position (station) in a certain time window, i.e., there exists $t \in [t_a^i - \epsilon_a^i, t_a^i + \epsilon_a^i]$ such that

$$s(t) = s^i \text{ and } u(t-0) < 0,\ v(t) = 0, \tag{7}$$

  or leaving of a position in a certain time window, i.e., there exists $t \in [t_l^i - \epsilon_l^i, t_l^i + \epsilon_l^i]$ such that

$$s(t) = s^i \text{ and } u(t+0) > 0,\ v(t) = 0, \tag{8}$$

  where $\epsilon_a^i$ and $\epsilon_l^i$ denote parameters that describes the allowed deviation from the planned schedule in a certain region of the network and $s^i$ as in (1).

In order to simulate the behaviour of the train on its journey over a given route through the network, in principle, we would need to know the position and velocity of the train as initial conditions as well as the control strategy.

Unfortunately, currently the only information (measurement) that is available from train movement is the position of the particular train at certain times $\bar{t}$, when the train passes detecting devices, i.e., we have (outputs)

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t), \quad \text{for } t = \bar{t}, \tag{9}$$

but we typically do not have the velocity of the train at this time instance.

It is clear that there are many strategies to design controls that yield a certain specified dynamics.

Typical criteria to design such a control are

- time optimal control,
- energy optimal control,
- following of a reference trajectory,
- minimal deviation from a desired schedule and
- mixed time and energy optimal control.

In this paper we will concentrate on time optimal control because this models a standard assumption on the driving strategy. For this the path of the train will first be partitioned into intervals where the velocity constraint is a continuous function. We allow jumps of the velocity constraints at the end points of these intervals. Stopping points (like stations) will also be end points of intervals.

## 3 Time optimal control for one train

In this section we introduce an approach for time optimal control of a single train. The motivation for this is the typical strategy of the engineer to always use the maximal allowed velocity as well as extremal acceleration or braking forces, respectively. Because, in general, schedule times are not calculated with technically shortest driving times, this allows for the engineer to compensate some delays. This assumption defines a well defined system and allows the determination (from knowledge of the initial position and velocity at certain time instances $t_j$) of the time the train needs to reach another position with a specified velocity.

We can formulate this for a single train as follows:

Consider the path of a train subdivided in intervals as in Figure 1 with starting point $s^0$ and end point $s^n$.

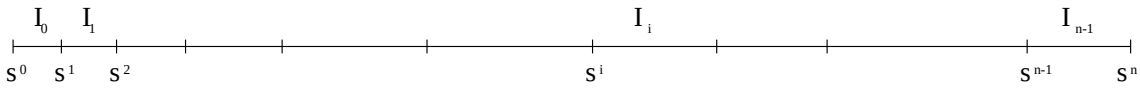


Figure 1: Subdivision of the line

The choice of this subdivision is induced from the simplifying assumption, that some data of the problem (esp. slope and velocity restrictions) are constant on these intervals. But other reasons for subdivisions are also possible.

The time optimal control problem is then to minimize $t_n - t_0$ subject to

$$\begin{aligned} \dot{x}(t) &= Ax(t) + B[u_f(s,v) + u_v(t)], \\ x(t_0) &= \begin{bmatrix} s^0 \\ v^0 \end{bmatrix}, \; x(t_n) = \begin{bmatrix} s^n \\ v^n \end{bmatrix}, \\ u_v(t) &\in [u_{min}(v(t)), u_{max}(v(t))], \end{aligned} \tag{10}$$

with further constraints given by (6). Time window restrictions as (7) and (8) will be omitted for the consideration here on time optimal control, because they model a more schedule-based driver strategy.

In this situation one is looking for a control $u(t)$, so that the movement of the train is a function $s = s(t)$ that connects the point $\left[\begin{smallmatrix} t_0 \\ s^0 \end{smallmatrix}\right]$ with the point $\left[\begin{smallmatrix} t_n \\ s^n \end{smallmatrix}\right]$ (see Figure 2) and $t_n - t_0$ is to be chosen minimal. The constraints then yield that

- the function $s(t)$ is monotonically nondecreasing, since $v(t) \geq 0$;
- the derivative $\dot{s}(t) = v(t)$ is not larger than the maximally allowed velocity.

To determine the time-optimal control, it is not relevant how long the train has to wait at a station (due to a constraint) or when it restarts after a stopping point, this just leads to an offset in the total time, therefore waiting times will be always 0 for simplicity here. A model incorporating waiting times is gained through a picewise time-shift of the relative times in such a zero-waiting-time model.

Combining the observations, we obtain the optimal control problem to minimize $t_n - t_0$ subject to

$$\dot{x}(t) = Ax(t) + B[u_f(s,v) + u_v(t)],$$

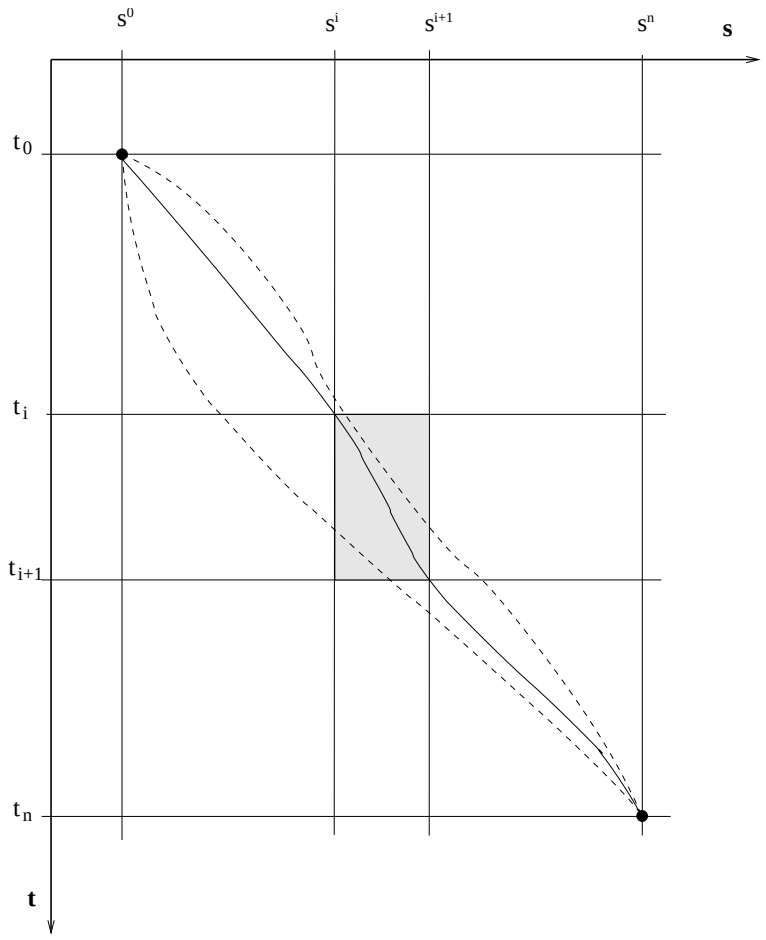


Figure 2: Time-path-lines of a train in the s-t-plane

$$\begin{aligned} x(t_0) &= \begin{bmatrix} s^0 \\ v^0 \end{bmatrix}, \\ x(t_n) &= \begin{bmatrix} s^n \\ v^n \end{bmatrix}, \\ v(s(t)) &\in [0, v_{max}(s(t))], \\ u_v(t) &\in [u_{min}, u_{max}(v(t))]. \end{aligned} \tag{11}$$

Using the classical theory of optimal control [5] we can reformulate this problem as the problem to minimize a cost functional

$$J(u) = \phi(x(t_f), t_f) + \int_{t_0}^{t_f} f^0(t, x, u)\ \mathrm{d}t \tag{12}$$

subject to constraints. Here $\phi(x(t_f), t_f)$ describes an optimization criteria at the end point and $f^0(t, x, u)$ is the cost function with respect to state and control at a certain time $t \in [t_0, t_f]$. In case of time optimal control $\phi(x(t_f), t_f) \equiv 0$ and $f^0(t, x, u) \equiv 1$. These constraints are first

of all equality constraints

$$\begin{aligned} \dot{x} &= f(x,u) \qquad f:\mathbb{R}^2\times\mathbb{R}^k\to\mathbb{R}^2, \\ x(t_0) &= x^0\in\mathbb{R}, \\ \Psi(x(t_f),t_f) &= 0, \quad \Psi:\mathbb{R}^2\times\mathbb{R}_+\to\mathbb{R}^q, \end{aligned} \tag{13}$$

withe some given function $\Psi$. Here $t_f \in \mathrm{R}_+$ is free, $q \leq n$ and the term $\phi(x(t_f), t_f)$ in $J$ is a weighting of the final state, which implies that in $\phi$ only those $2-q$ components of $x$ should be weighted that are not fixed in $\Psi$.

Furthermore, we may have inequality constraints for the controls and for the states

$$C(x(t),u(t),t)\leq 0 \qquad C:\mathbb{R}^2\times\mathbb{R}^k\times\mathbb{R}_+\to\mathbb{R}^l \tag{14}$$

as well as extra conditions at interior points

$$N_j(x(t_{i_j}),t_{i_j})=0 \qquad N:\mathbb{R}^n\times\mathbb{R}_+\to\mathbb{R}^q,\ i_j\in\{0,1,\ldots,n\},\ j=1,\ldots,p$$

If we assume that all conditions are sufficiently often differentiable then we can define the following *Hamiltonian*

$$H(x,u,\lambda)=f^0(t,x,u)+\lambda^T f(x,u) \tag{15}$$
$$\lambda:[0,t_f]\to\mathbb{R}^2$$

as well as an auxillary function

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

that connects fixed end points with free end points that will be weighted in the cost functional.

We then obtain, see [5], the following necessary conditions for an optimal control

$$\begin{aligned} H_x(\bar{x},\bar{u},\bar{\lambda})(x-\bar{x}) &\geq 0 \qquad \forall x(\cdot), \\ H_u(\bar{x},\bar{u},\bar{\lambda})(u-\bar{u}) &\geq 0 \qquad \forall u(\cdot), \end{aligned}$$

where $H_x$ and $H_u$ denote the partial derivatives with respect to $x$ and $u$, respectively.

In this way the optimal control problem is turned into a boundary value problem that we could solve using classical methods for boundary value problems.

But to determine the solution of this boundary value problem it is necessary to determine the switching points for the control that determine the inhomogeniety and hence the solution, see [5]. If these switching points are known, then we have a multi-point boundary value problem that we could approach by finite-element, finite difference or multiple-shooting methods [1].

Once these switching points have been determined, however, it is well known [5] that the time-optimal control is given by a bang-bang control, where the control always switches between its extremal values in the constraints.

This simplifies the solution in the time-optimal case significantly and we will use this approach here. But we should note that for all other optimality criteria it is necessary to solve a boundary value problem. This will be the topic of further investigations.

# 4 Calculation of switching points

In this section we discuss the numerical computation of switching points. There are three types of possible switching points:

- points where the velocity or the acceleration of the train reaches the boundary of the constraints interval, while not being on the constraint before;
- points where the boundary of constraint intervals is left;
- points where the control switches from acceleration to braking to meet other constraints.

The computation of the switching points is done by determining the intersections of the trajectories of the velocity constraints and the trajectories obtained with extremal control.

The solution of this problem is quite simple if the following assumptions hold:

**Assumption 1** *a) Constraints in one path interval do not have any influence on the control in previous intervals. This condition would assure that the points where the control leaves the extremal value can be determined successively backwards from the following block. Figure 3 shows a situation where this assumption is violated.*

*b) The increase in the velocity constraint is not too large, so that the control is able to follow it until the next possible point, where braking has to set in, see Figure 4. This would ensure that in every block there is at most one point where the constraint is reached and one where the constraint is left.*

*c) When the velocity constraint is decreasing, then the control of maximal braking is able to follow this constraint.*

For simplicity we, furthermore, assume that in every interval the velocity constraint is given by a piecewise polynomial of degree at most 3,

$$v_{max}(s) = p_0 + p_1 s + p_2 s^2 + p_3 s^3. \tag{16}$$

If these assumptions hold, then we have the following methods to determine the switching points.

## 4.1 Reaching the constraint

We consider first the situation that the current velocity in a path point $s^i$ is smaller than the constraint $v_{max}(s^i)$ and assume that we know the time instance $t_i$ at which the train is in the point $s^i$, the current velocity $v(t_i)$ as well as the control $u_v = u_{max}(v(t_i))$.

We want to determine the time $t_i^r$ for which $v_i^r := v(t_i^r) = v_{max}(s(t_i^r))$. Here we assume that $s(t_i^r) \leq s^{i+1}$, otherwise we are not able to reach the constraint. We define the monotone function

$$\sigma(t) := v_{max}(s(t)) - v(t) \tag{17}$$

and obtain the time instance, where the constraint is reached, i.e., when $\sigma(t)$ is zero. Thus, we have a root finding problem that can be solved by classical root-finding methods [2, 8], like the bisection method, Newton's method or fix-point iterations. A comparison of different methods will be the topic of further investigations, here we concentrate on the following bisection method.

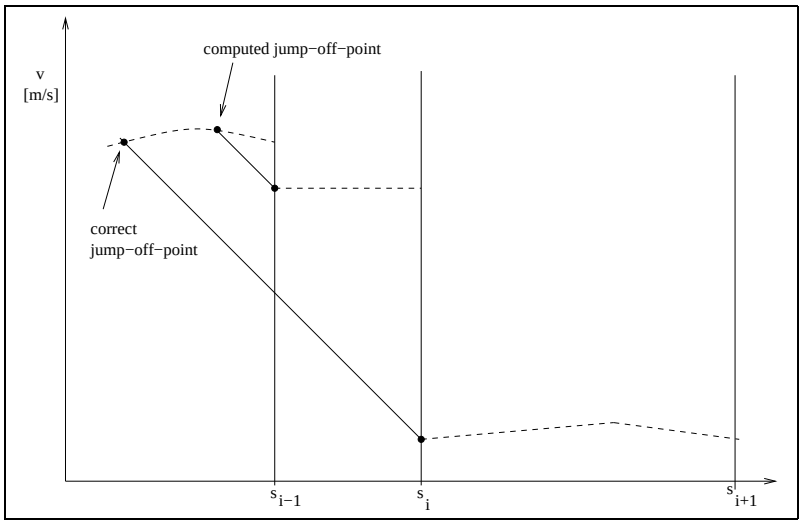


Figure 3: Assumption 1a) for velocity constraint

**Algorithm 1** *Bisection method to compute the time instance where the velocity constraint is reached.*

**Input:** *Tolerance for accuracy $\epsilon$, starting stepsize $\Delta t$ and initial values $t_i$, $s^i$ and $v^i$.*

**Output:** *Numerical value for time instance $t_i^r$ and for location $s_i^r$, where the constraint is reached.*

1. *Set $\hat{t}_0 = t_i$ and $x_0 = [s^i, v^i]^T$.*
2. *Set $\hat{t}_1 = \hat{t}_0 + \Delta t$.*
3. *Determine $\sigma(\hat{t}_1)$ (17) by solving the initial value problem*
$$\begin{aligned} \dot{x}(t) &= Ax(t) + B[u_f(x_1(t), x_2(t)) + u_{max}(x_2(t))],\ t \in [\hat{t}_0, \hat{t}_1] \\ x(\hat{t}_0) &= x_0. \end{aligned} \tag{18}$$
4. *If $|\sigma(\hat{t}_1)| < \epsilon$ then set $t_i^r := \hat{t}_1$, $s_i^r := x_1(\hat{t}_1) = s(\hat{t}_1)$ and STOP.*
5. *If $\sigma(\hat{t}_1) < 0$, then half the stepsize $\Delta t := \frac{1}{2}\Delta t$.*
6. *If $\sigma(\hat{t}_1) > 0$, then we have not reached the constraint yet. Set $x_0 = x(\hat{t}_1)$, $\hat{t}_0 := \hat{t}_1$.*
7. *GOTO Step 2.*

For the numerical implementation of the bisection method we note the following:

1. If the constraints cannot be reached in the given interval, then we have to check whether it is possible that the computation of the switching point $t_i^r$ leads to $s_i^r > s^{i+1}$. In this case we move through the interval with maximal velocity.
2. It is possible to choose the parameters so that the optimal control becomes $u = u_f + u_{max} = 0$. This is the case if the train is in a very steep climb. In this case the control is assumed to stay 0 and hence the train stays (under the assumption that the braking force can prevent it from going backwards). We then stop the bisection method with an error message.

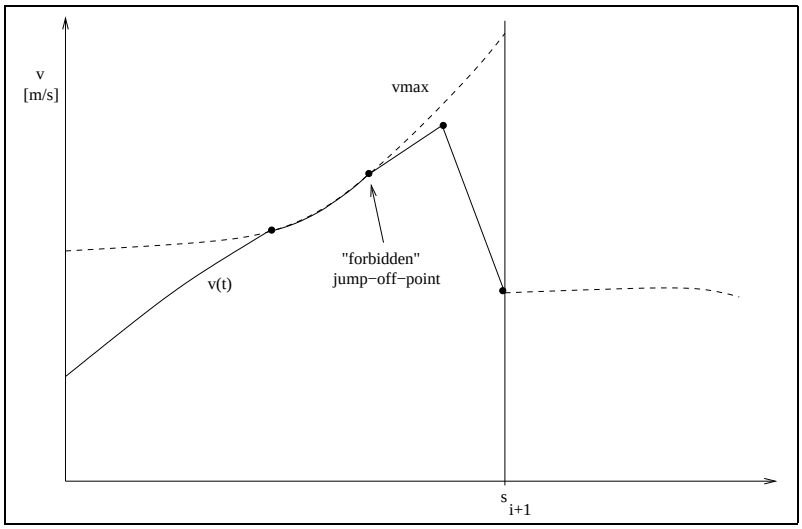


Figure 4: Assumption 1b) for velocity constraint

### 4.2 Leaving the constraint

If the velocity constraint drops at the end of a block, i.e., if $v_{max}(s^{i+1}+0) < v_{max}(s^{i+1}-0)$, then it is necessary to leave the constraint earlier and to determine the switching point $(t_i^l, s_i^l)$, where the train has to leave the constraint.

In this case we run the computation backwards and start at $s^{i+1}$ with the maximal velocity limit from the right, moving with a variable negative control. Again we use a bisection method to do the root-finding. As starting values we may use $t = 0$ and $s = 0$. Via the solution of the initial value problem

$$\begin{aligned} \dot{x}(t) &= Ax(t) + B[-u_f(x_1(t), x_2(t)) - u_{min}(x_2(t))], \\ x(0) &= \begin{bmatrix} 0 \\ v_{max}(s^{i+1}+0) \end{bmatrix}, \end{aligned} \tag{19}$$

we obtain $s(t) = x_1(t)$ the length of the path from $s^{i+1}$ to the switching point. To determine $v_{max}$, however, we need the absolute path, hence we determine $v_{max}$ as $v_{max}(s^{i+1} - s(t))$. Apart from this change we may use Algorithm 1 with (19) instead of (18) and we determine the position $s_i^l$ of the switching point, the velocity at the switching point (which is the maximal velocity in this point) and the necessary absolute time $t_b$ needed for the braking to meet the constraint in the following interval. The real time instance $t_i^l$, where the switching takes place is determined later. Note that Assumption 1 guarantees that always a switching point $s_i^l$ exists in the given interval $[s^i, s^{i+1}]$.

Let us assume now that $s_i^r \le s_i^l$. This implies that also $t_i^r \le t_i^l$, since $s(t)$ is a continuous and monotonically non-decreasing function.

We still need to determine the time period that will pass while being on the velocity constraint, i.e., $t_i^l - t_i^r$. Again we use a root-finder to determine this period as a root of the function

$$\sigma(t) := s(t) - s_i^l, \tag{20}$$

where $s(t)$ is the solution of the initial value problem

$$\begin{aligned} \dot{s}(t) &= v_{max}(s(t)), \\ s(t_i^r) &= s_i^r. \end{aligned}$$

This corresponds to using the velocity $v_{max}(s(t))$ in the interval $[s_i^r, s_i^l]$.

Let the root of $\sigma$ be given by $t_d$. Then we obtain the time point where we have to leave the constraint as $t_i^l := t_i^r + t_d$ and the total travel time in the $i$-th block as the sum of the time to reach the constraint, the time $t_d$ on the constraint and the time to reach the end point of the block, i.e., $t_i^r - t_i + t_d + t_b$.

Our modelling concept realizes the safety concept for the interaction of the different trains through the creation of alternative velocity constraints which are switched on and off dynamically during the simulation. In other words every train is either constrained by hard velocity constraints which arise from the infrastructure and can be precomputed off-line before the simulation or by velocity constraints set by the other trains during the simulation. For this reason it is sufficient to determine the points where velocity constraints are reached or when the train has to leave the constraint. The third possibility that a train has to start braking before it even reaches the constraint cannot happen after the hard velocity constraints have been computed.

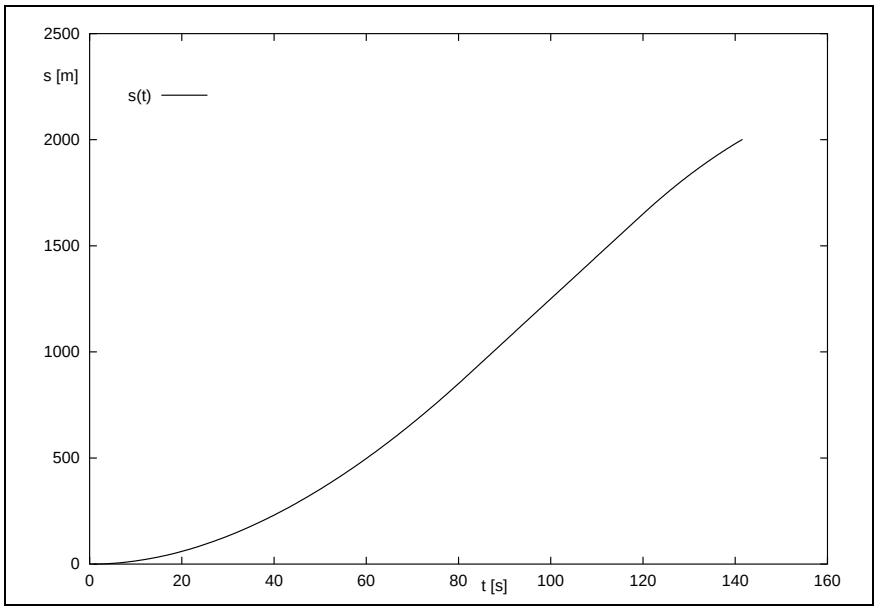


Figure 5: s-t diagram for problem, where constraint is reached.

In Figures 5 and 6, we depict the trajectories $s(t)$ and $v(t)$ for an example where the constraint is reached.

# 5 Problems without Assumption 1

Unfortunately the intervals in realistic train networks can have very small length, so that Assumption 1 cannot always be guaranteed. For this reason we propose a different strategy for the determination of the points where to leave the constraints.

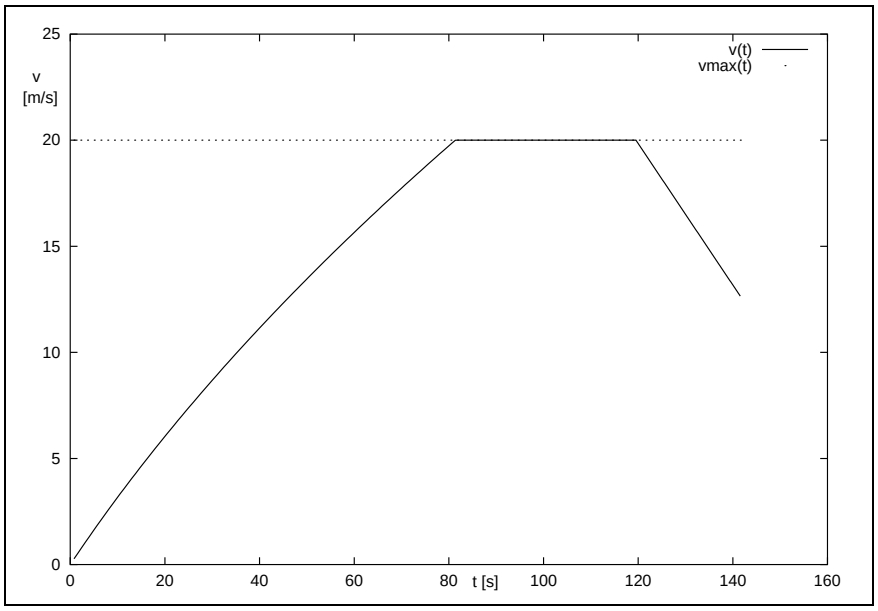


Figure 6: v-t diagram for problem, where constraint is reached.

We build a list with all right velocity constrains $v_{max}(s_i + 0)$ at all end points $s_i$ of the block and store the list in $\mathtt{v_{maxr}[i]}$.

Then we go backwards through all intervals of the whole train line to classify whether the intervals have *normal* lengths or are too *short*.

We begin in the last point of the whole line and determine the point $s_n^l$, where the constraint has to be left in the last interval. If this point is in the interval, then we mark this interval as *normal* and proceed to the second last interval and so forth. If it is not possible to reach the point where we leave the constraint in the interval, we determine the actual velocity $v^*$ in $s^{i-1}$, and correct the list of maximal right velocities at an interval boundary by setting $\mathtt{v_{maxr}[i-1]}= v^*$. Then we mark this interval as *short* and proceed.

In this way we know that in a short interval the velocity constraint is given by the braking curve and in all normal intervals Assumption 1a) is valid. See Figure 7.

# 6 A special case

In this section we discuss the time optimal control in a realistic situation, where several simplifications hold. This special case, that was considered in a project of the last author at TLC GmbH, is as follows:

- Within one interval the maximally allowed velocity is a constant function of the position.
- The forces that describe the acceleration and braking process are constant functions of the position and continuous functions of the velocity.
- At the end points of blocks jumps in the velocity constraints and the exterior forces are allowed.

Under these simplifications the computation of switching points simplifies significantly even without Assumption 1. We again have to determine the points where the velocity constraint is reached and left.

### 6.1 The adaption of velocity constraints

Due to the fact that the negative acceleration $a$ during the braking process is assumed constant, we do not have to solve a differential equation, but merely an integration is sufficient.

If we are in a block $i$ at a possible switching point $s_i^s$ with velocity $v_i^s$ and start at time $t_0 = 0$ then we have

$$\begin{aligned} v(t) &= at + v_i^s, \\ s(t) &= \frac{a}{2}t^2 + v_i^s t + s_i^s. \end{aligned}$$

In order to achieve a velocity $v(\tilde{t}) = \tilde{v}$ by braking, we obtain a necessary time

$$\tilde{t} = \frac{\tilde{v} - v_i^s}{a}$$

and a switching point

$$s_i^s = \tilde{s} - \frac{a}{2}(\tilde{t})^2 - v_i^s \tilde{t} \text{ with } \tilde{s} = s(\tilde{t}).$$

Usually we have $\tilde{t} = t_{i+1}$, $\tilde{s} = s_{i+1}$ and $\tilde{v} = v_{max}(s_{i+1} + 0)$.

If $s_i^s > s_i$ then we have found a point where we have to leave the velocity constraint, see Figure 6.

If the block is *short*, i.e., $s_i^s \leq s_i$ then we have to correct the maximal velocity at the beginning of the block to the velocity $v^*$ that we can maximally allow to reach the maximally allowed velocity at the end of the block, see Figure 7.

Using the equations of motion

$$\begin{aligned} v(t) &= at + v^*, \\ s(t) &= \frac{a}{2}t^2 + v^* t + s_i. \end{aligned}$$

we obtain $v^*$ in terms of the unknown travel time

$$v^* = \tilde{v} - a\tilde{t}. \tag{21}$$

Then with

$$\tilde{s} = \frac{a}{2}\tilde{t}^2 + (\tilde{v} - a\tilde{t})\tilde{t} + s_i$$

we obtain the necessary time for braking from the quadratic equation

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

with the solutions

$$\tilde{t}_{1,2} = \frac{\tilde{v}}{a} \pm \sqrt{\frac{\tilde{v}^2}{a^2} + \frac{2(s_i - \tilde{s})}{a}}.$$

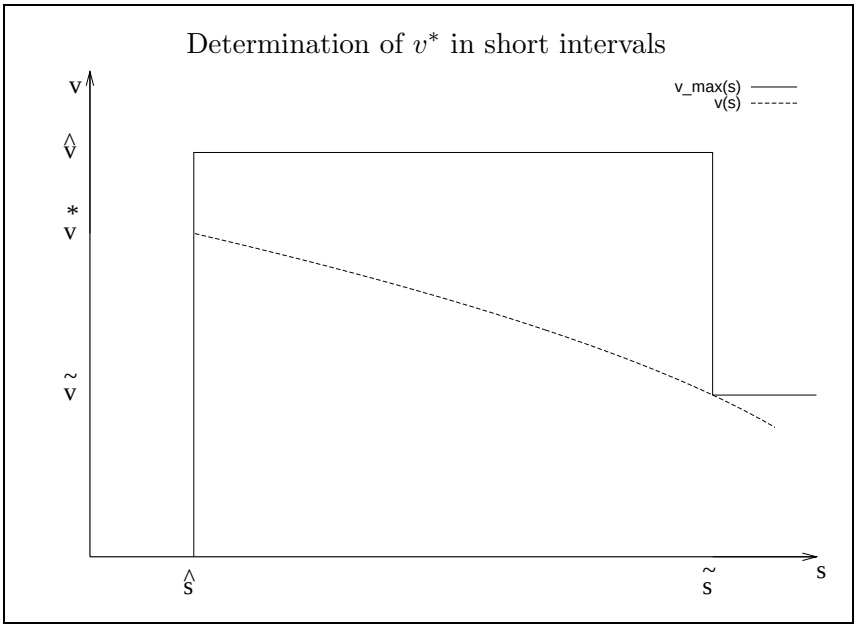


Figure 7: Correction of $v_{max}$ in short blocks

A short calculation yields that the relevant solution is given by

$$\tilde{t} = \frac{\tilde{v}}{a} + \sqrt{\frac{\tilde{v}^2}{a^2} + \frac{2(s_i - \tilde{s})}{a}}$$

and we have determined $v^*$ by (21). We enter this into the list and note that in this interval full braking is necessary.

It should be noted that in intervals with full braking we can determine in a similar fashion the maximal velocity, see Figure 8. The velocity constraint in such an interval with left maximal velocity $v_i$ is then given by

$$\begin{aligned} v_{max}(s) &= a\left[-\frac{v_i}{a} + \sqrt{\frac{v_i^2}{a^2} - \frac{2(s_i - s)}{a}}\quad\right] + v_i \\ &= -v_i + a\sqrt{\frac{v_i^2}{a^2} - \frac{2(s_i - s)}{a}} + v_i \\ &= a\sqrt{\frac{v_i^2}{a^2} - \frac{2(s_i - s)}{a}} \\ \Leftrightarrow v_{max}(s) &= \sqrt{v_i^2 - 2a(s_i - s)}. \end{aligned} \tag{22}$$

In the numerical implementation it may happen that due to roundoff errors we obtain small complex values. To guarantee that this does not happen we use

$$v_{max}(s) = \sqrt{|v_i^2 - 2a(s_i - s)|}.$$

In Figures 9 and 10 we depict the original and the corrected $v_{max}(s)$. In the following we use $v_{max}(s)$ to denoted the corrected maximal velocity.

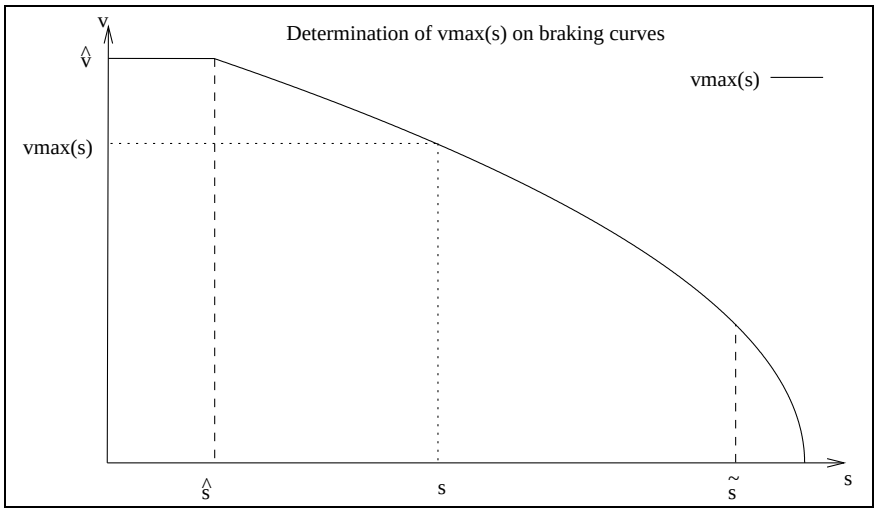


Figure 8: $v_{max}(s)$ as braking curves

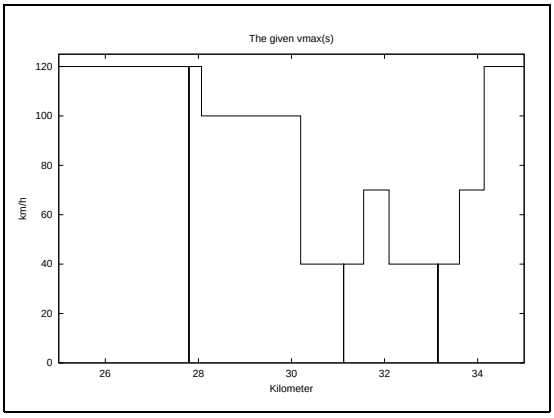


Figure 9: $v_{max}(s)$ before correction

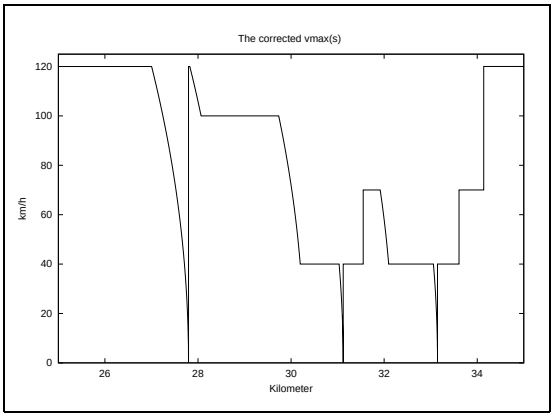


Figure 10: $v_{max}(s)$ after correction

## 6.2 The simulation process for one train

We also have several simplifications in the simulation process. If we start at a boundary point of a block with a velocity equal to the maximal velocity and if $u_{max}(v) \geq -u_f(s,v)$, then we obtain for $v_{max}(s) = const$ a travel time of $t = \frac{\texttt{intervallength}}{v_{max}}$. If $v_{max}(s)$ is not constant, then due to the constant acceleration we get $t = \frac{v_{new}-v_{old}}{a}$, where $v_{old} = v_{max}(s)$ und $v_{new} = v_{max}(s + \texttt{intervallength})$.

In both cases the initial velocity in the next block is given by the maximal velocity on the left side of the next block.

If the initial velocity is not on the upper constraint, i.e., $v \in [0, v_{max})$, or if $u_{max}(v) < -u_f(s,v)$ , then we solve the system

$$\begin{aligned} \dot{s} &= v, \\ \dot{v} &= \frac{1}{m}(u_{max}(v) + u_f(s,v)), \end{aligned}$$

$$\begin{aligned} s(0) &= s_0, \\ v(0) &= v_0 \end{aligned}$$

until we go beyond the right boundary of the block or until $v(s(t)) \geq v_{max}(s(t))$.

As numerical integration methods we have tested the forward Euler-method with constant and variable stepsize as well as a fourth order Runge-Kutta-method with a simple extrapolation based stepsize control [8, 2].

If one of the stopping criteria (right boundary or maximal velocity) is reached then we used linear or polynomial interpolation of the computed values to determine the values at the boundary or we used Newton's method to determine the points where we reach the constraint.

A special treatment is necessary for stopping points (like in a station) which have zero interval length or intervals where the slope of the track is too large so that the end of the interval cannot be reached.

### 6.3 Numerical examples

In the following we leave off measurement units. The data are measured in kg, N, m, m/s, m/s$^2$, ....

As a first test case we consider a hyperbolic accelerating force. Consider the system

$$\begin{aligned} \dot{s} &= v, \\ \dot{v} &= \frac{1}{m} u(v), \end{aligned}$$

where

$$u(v) = \frac{250000}{v}.$$

We obtain the system

$$\begin{aligned} \dot{v} &= \alpha \frac{1}{v} \qquad \text{with} \quad \alpha := \frac{250000}{m}, \\ v\dot{v} &= \alpha, \qquad v\dot{v} = \frac{1}{2}(\dot{v^2}), \quad w := v^2, \\ \dot{w} &= 2\alpha, \\ w &= 2\alpha t + C_1. \end{aligned}$$

Hence

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

and with

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

we obtain

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

With initial values $s(0) = 0$ and $v(0) = 1$ we get $C_1 = 1$ and $C_2 = -\frac{2}{6\alpha}$.

With a mass $m$ of the train of 500 000 kg, we get $\alpha = \frac{1}{2}$ and

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

Starting in $t = 0$ with $s(0) = 0$ and $v(0) = 1$, we reach the maximal velocity of 30 m/s after 899 s and 17999.33 m.

$$\begin{aligned} v(899) &= \sqrt{899+1} &&= 30, \\ s(899) &= \tfrac{2}{3}(899+1)^{\frac{3}{2}} - \tfrac{2}{3} &&= 17999.33. \end{aligned}$$

In an analogous way we constructed another example with an interval of length 5332.67m in which no constraint is reached and where the interval is left after $t = 399$ s with velocity $v = 20.0$ m/s.

Some results of the numerical tests are given in Section 11.1. Another testcase used realistic data of the network of Deutsche Bahn and is referenced as testcase 22127-1013, see Section 11.2.

# 7 Several interacting trains

In this section we discuss the simulation of several ($n_z$) trains that interact with each other via a security concept (e.g. a fixed block signal system) which prevents a train from entering in a certain block until any other train has left this block.

In the following we assume that the path that a specific train takes through the network is fixed a priori.

The path of the train with index $i$, where $i = 1, \ldots, n_z$ is given by an interval $L^i := [s_0^i, s_f^i] \subset \mathbb{R}$. Associated with this path are the following piecewise continuous functions (which in the special case of Section 6 are piecewise constant):

- the maximal velocity $v_{max}^i : L^i \to \mathbb{R}_0^+$,
- the slope of the track $\omega^i : L^i \to \mathbb{R}$.

For every train there is also given a maximal negative (braking) acceleration $a_{min}^i \in \mathbb{R}^-$.

The state of each train is given again as

$$x^i(t) := \begin{bmatrix} s^i(t) \\ v^i(t) \end{bmatrix},$$

where $s^i(t)$ is the position and $v^i(t)$ is the velocity of the $i$-th train at time $t$.

We now describe a mathematical model for the safety concept (which is in accordance with a fixed block safety concept). For this concept the paths $L^i$, of all trains $i = 1, ..., n_z$ are divided into $m_s^i$ *safety blocks* $S_j^i \subset L^i$, $j = 1, \ldots, m_s^i$. This partitioning into blocks is at first independent of the subdivision depicted in Figure 1.

It should be noted that every individual train has its own string of safty blocks. It may happen that different trains use (in some order) the same physical piece of track but then this block has different labels for the different trains. One duty of the safety concept via velocity constraints described below is to have a modeling concept at hand which incorporates the

information which trains may not use the same physical blocks at the same time as data into the simulation procedure.

In a classical signal system with pre- and main signal (see e.g. [6]) every block extends from a *pre-signal* to the overnext *main-signal*, because

1. this is the region which a train uses exclusively under normal operating conditions and
2. at the end of such a block the train stops if "Stop" is signalled.

In other fixed block safety systems (e.g. LZB) a similar choice of such blocks is also possible with the same criteria.

The safety concept is described using *braking curves*

$$b_j^i : S_j^i \to \mathbb{R}_0^+ \tag{23}$$

that are monotonically decreasing functions which satisfy $b_j^i(s) = 0$ at the right boundary of $S_j^i$. These curves act as state-dependent velocity constraints. In general they may be chosen as linear functions with slope $a_{min}^i$. Switching these braking curves on and off - this is equivalent to switching the related signal between "Stop" and "Go" - by position of all the trains in consideration realizes the safety concept. This does not model the full behavior of the signal system, but is the correct intrinsic mechanism induced from other trains.

Furthermore, for every safety block we have an *indicator function*

$$\chi_j^i : \mathbb{R}^{n_z} \to \{0, 1, 2, ...\}, \tag{24}$$

that satisfies

$$\chi_j^i(s^1, \ldots, s^{n_z}) \quad \begin{cases} = 0 & \text{if the train } i \text{ may leave block j,} \\ > 0 & \text{if one of the other trains prevents the train } i \text{ from leaving block } j. \end{cases} \tag{25}$$

This means that in case that $\chi_j^i > 0$, the train $i$ has to stop at the end of the block $S_j^i$. More precisely, the positive values of $\chi_j^i$ are chosen as the number of conditions preventing the train from leaving block $S_j^i$; this may be other trains or a priori set of conditions. These indicator functions may change their values only if a train enters or leaves a safety block. It should be noted that these $\chi_j^i$ depend in fact only on the position variable $s^i$ in $x^i$.

If a train enters a safety block then for all other associated blocks (these are the safety blocks of other trains that use this physical piece of track) the indicator function is increased by 1, i.e., either the main signal is switched to "Stop" or stays in this position.

If a train leaves a safety block then the value of the indicator functions of the other associated blocks is reduced by 1 if it was positive.

Since a safety block extends from a pre-signal to the overnext main-signal it is clear that every train is at least in one and at most in two safety blocks, see Figure 11.

In order to simplify the model in the case the train is only in one of these blocks we introduce a *dummy-block* $S_0$ with the property that

$$\chi_0(s^1(t), \ldots, s^{n_z}(t)) = 0 \quad \forall t,$$

i.e., in such a block a braking curve is never active. Inserting this dummy-block, if necessary, we can achieve that every train is always exactly in two safety-blocks, see Figure 12.

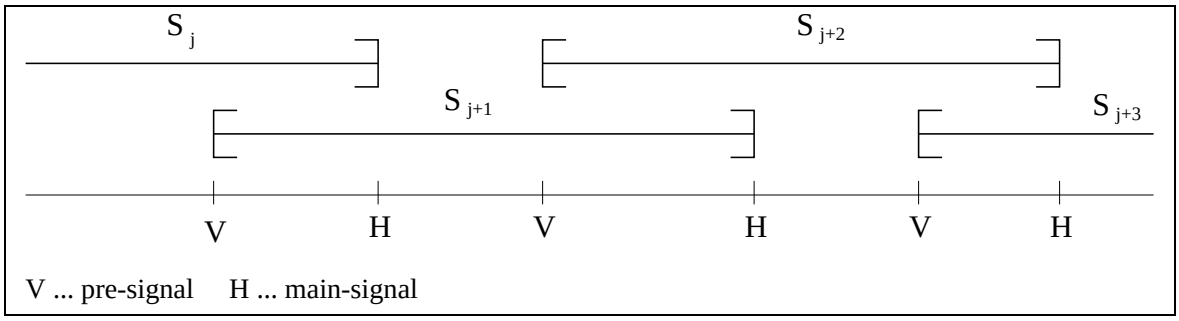


Figure 11: Distribution of safety blocks of one train over its path

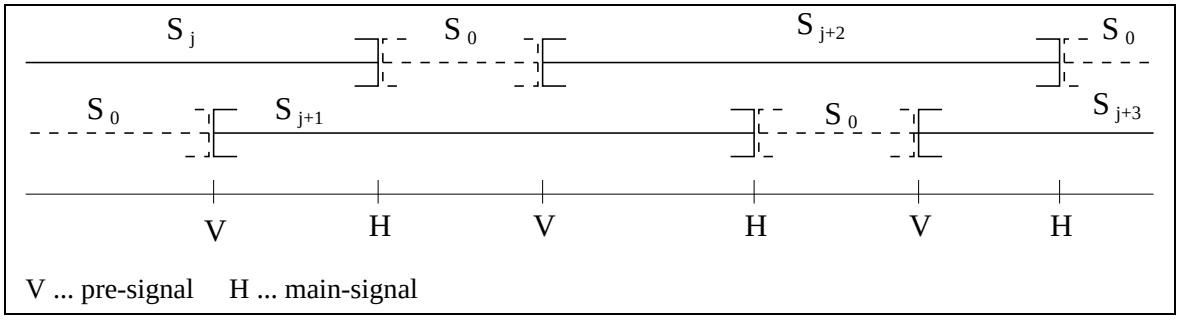


Figure 12: Inclusion of dummy-blocks

These two safety-blocks will be locally labeled with 1 and 2 and then using the functions

$$\sigma^i : \{1,2\} \times L^i \to \{0,1,\ldots,m_s^i\} \tag{26}$$

the mapping of the local labels to the global labels of the momentary position of train $i$ is realized.

In this way there exist three locally active velocity restrictions for every train, the maximal velocity $v^i_{max}(s^i)$ as well as those given by the braking curves, i.e., velocities achieved with maximal braking

$$v^i_{max1}(t,s^i) := \begin{cases} b^i_{\sigma(1,s^i)} & \text{if } \chi^i_j(s^1(t),\ldots,s^{n_z}(t)) > 0 \\ +\infty & \text{otherwise} \end{cases},$$

$$v^i_{max2}(t,s^i) := \begin{cases} b^i_{\sigma(2,s^i)} & \text{if } \chi^i_j(s^1(t)\cdots s^{n_z}(t)) > 0 \\ +\infty & \text{otherwise} \end{cases}.$$

The relevant restriction $v^i_{rel}(t,s^i)$ for all trains $i, i = 1,\ldots,n_z$ is then the minimum of these 3 restrictions

$$v^i_{rel}(t,s^i) := \min\{v^i_{max}(s^i), v^i_{max1}(t,s^i), v^i_{max2}(t,s^i)\}. \tag{27}$$

The indicator functions $\chi^i_j(s^1(t),\ldots,s^{n_z}(t))$ are defined specifically to describe the number of active blockings of the safety block $j$ of train $i$ at time $t$. If $\chi^i_j(s^1(t),\ldots,s^{n_z}(t)) = 0$ then this block is open and a train can leave this block. If $\chi^i_j(s^1(t)\cdots s^{n_z}(t)) = k$ with $k \neq 0$, then

there exist $k$ blockings. Typically (e.g. if two trains follow each other), then $k = 1$, but there exist circumstances, when $k$ may be larger (e.g. if one train has to let several other trains pass near a crossing). Every safety block is associated with exactly one train. Because of the monotonicity and continuity of the $s^i(t)$ the values of $\chi^i_j(t)$ may change only at times where a train enters resp. leaves a block.

A more systematic way to define $\chi^i_j(t)$ is therefore to describe its changes. This can be modelled via a *point-set map*.

**Definition 1** *Let $M$ be a set and $\mathbf{P}(M)$ the power set of $M$. A mapping $F : M \to \mathbf{P}(M)$ is called point-set map if every $m \in M$ is mapped to some subset $F_m \subset M$.*

Let $\beta^i \in \mathbb{N}_0^{m^i_s}$ be the vector with the number of blockings for all safety blocks related to train $i$ at a certain time. At initial time $t_0$ this vector will be initialized with values $\beta^i_j \in \mathbb{N}_0$.

Furthermore we have the point-set maps

$$\begin{aligned} \texttt{block} &: \{1, ..., n_z\} \times \mathbb{N} \to \mathbf{P}(\{1, ..., n_z\} \times \mathbb{N}), \\ \texttt{unblock} &: \{1, ..., n_z\} \times \mathbb{N} \to \mathbf{P}(\{1, ..., n_z\} \times \mathbb{N}), \end{aligned}$$

which are defined here on the set of indices for the blocks as the basic set. (Using the set of blocks itself as the basic set would lead to an equivalent definition.)

In a preprocessing step these point-set maps are stored in a table representing the relevant information of the interconnections in the network and some extra data. As an example see Tables 1 and 2.

If train $i$ enters a block $j$ at some time, we can determine the blocking conditions of the relevant other safety blocks using these tables from the old values of $\beta^k_l$ as

$$\beta^k_l := \beta^k_l + 1 \qquad \forall (k, l) \in \texttt{block}(i, j).$$

Analogously, if train $i$ leaves a block $j$ at some time, we obtain the unblocking of other safety blocks as

$$\beta^k_l := \max\{\beta^k_l - 1, 0\} \qquad \forall (k, l) \in \texttt{unblock}(i, j).$$

Therefore at every time $t$ the value of $\beta^k_l$ represents the value of $\chi^k_l$ since the last event (time of block leaving or block entering of some train) where the value of $\chi^k_l$ may have changed. This means that $\chi^k_l$ is piecewise constant between such events and has jumps of integer heights (mostly from $\{+1, -1\}$) at these event times. It should be noted that the point-set map `block` may also be used to construct initial values $\beta^i_j$ from the initial positions $s^i(t_0)$ of all trains.

In the implementation this two dimensional indexing, e.g. $S^i_j$, $\chi^i_j$ or $\beta^i_j$, is stored in an one dimensional array, e.g. $S_k$, $\chi_k$ or $\beta_k$ with $k = (0,)1, ..., \sum_{i=1}^{n_z} m^i_s$.

# 8 Interaction between trains

In this section we discuss the modelling of different kinds of interactions between trains in a network. In order to do this we distinguish two types of blockings for certain intervals, blockings due to safety constraints and blockings that are related to the order in which trains are allowed to pass certain common parts of a network, like e.g. priority rules for different trains. These two kinds of blocking safety blocks of trains may be given as input data by appropriate initial settings of the vector $\beta := (\beta^i_j)_{i,j}$, the initial configuartion, and definitions of the point-set maps `block` and `unblock` as shown by the following prototypic examples.

## 8.1 Consecutive trains on one line

If one train follows another on the same track, in the model as well as in reality it has to be guaranteed that the second train does not catch up with the first one. There is some freedom in the choice of the used combinations of the initial configuration for $\beta$, the definitions of the `block/unblock` point set maps and the assumed minimal starting times of the two trains. The difference in these choices will not be in the effective behavior of the simulation but in the way the local phenomena of the signal system are represented.

We assume for this example that we have $m_s^i$ safety blocks for train $i$, $i = 1, 2$, that for both trains all safety blocks are the same in their position on the line and that $m_s^1 = m_s^2 =: m_s$. It follows that $S_j^1$ and $S_j^2$, $j = 1, \ldots, m_s$ denote the same physical pieces of the track, see Figure 13.

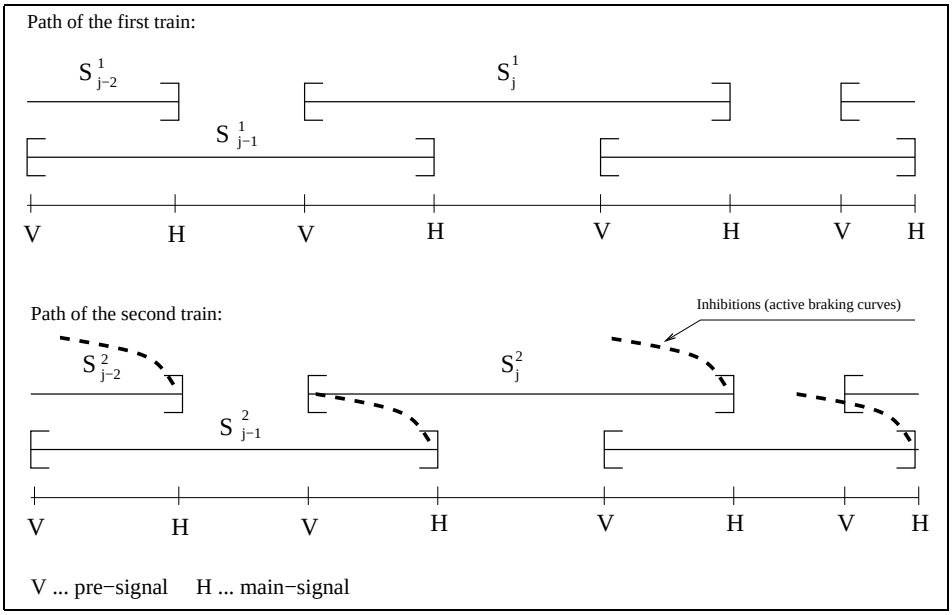


Figure 13: Two consecutive trains on the same line

First we will pursue that initially the whole line is blocked for the second train and the first train opens the blockings when it leaves appropriate safety blocks. Therefore we choose the following initial configuration for $\beta$:

$$\beta_j^i = \begin{cases} 0 & i = 1, \\ 1 & i = 2. \end{cases} \tag{28}$$

Consider the situation that the first train has just left the block $S_j^1$. According to the principle, that there has to be at least one red signal between two consecutive trains, the second train is allowed to use the complete block labeled $S_j^2$ (this is physically seen the same block as $S_j^1$ for the first train), which means that the braking curve in the block of the second train associated with $S_{j-1}^2$ can be switched off. Hence the first train unblocks $S_{j-1}^2$ as leaving the block $S_j^1$ and the point-set maps `block(i,j)` and `unblock(i,j)` have the following form:

$$\texttt{block}(i,j) \quad = \quad \emptyset \qquad i = 1, 2, \qquad j = 1, \ldots, m_s \tag{29}$$

$$\texttt{unblock}(i,j) \;=\; \begin{cases} \emptyset & i=1, \quad j=1, \\ \{(2,j-1)\} & i=1, \quad j=2,\ldots,m_s, \\ \emptyset & i=2. \end{cases} \tag{30}$$

The induced behavior here is independent of the initial states of the trains resp. their minimal starting times. The first train (using the blocks $S_j^1$) will travel always as first, because otherwise any block of the second train would remain blocked. The effect of this modelling is a kind of keeping some distance (here one block, but any number of blocks is possible) between the two trains, but it does not reflect the intrinsic mechanisms of the locally acting fixed block safety system.

The second model starts with the construction of the `block/unblock` maps to catch completely the local effects of the safety system independent of the actual sequence of trains. It is based on the principle cited above (at least one red main signal between two consecutive trains): When the first train is in block $S_j^1$ but not in $S_{j-1}^1$, then the second train is not allowed to enter the block $S_j^2$. This is guaranteed by activating the braking curve associated with $S_{j-1}^2$. As long as the train is in the intersection of $S_{j-1}^2$ and $S_{j-2}^2$, the braking curve in $S_{j-2}^2$ has to be active. These arguments are valid also by interchanging the two trains. Because of the choice of the blocks such that every train should use them exclusively, the unblocking is completely analogous to the blocking concept and we obtain the two identical point-set maps

$$\texttt{block}(i,j) \;=\; \begin{cases} \emptyset & i=1,2 \quad j=1, \\ \{(2,j-1)\} & i=1, \quad j=2,\ldots,m_s, \\ \{(1,j-1)\} & i=2, \quad j=2,\ldots,m_s, \end{cases} \tag{31}$$

$$\texttt{unblock}(i,j) \;=\; \begin{cases} \emptyset & i=1,2 \quad j=1, \\ \{(2,j-1)\} & i=1, \quad j=2,\ldots,m_s, \\ \{(1,j-1)\} & i=2, \quad j=2,\ldots,m_s. \end{cases} \tag{32}$$

With these (with respect to $i$) fully symmetric point set maps one gets a realistic behavor for any realistic choice of initial conditions: The initial configuration of $\beta$ should reflect the starting positions of the trains, e.g. if train 1 starts in block $S_1^1$ and train 2 starts in block $S_2^2$ at the same minimal starting time then

$$\beta_j^i = \begin{cases} 1 & i=j=1, \\ 0 & \text{otherwise.} \end{cases} \tag{33}$$

is a consistent choice of $\beta$, which leads to the desired simulation behavior, where train 1 follows train 2 and will never catch up. Interchanging the two trains or choosing different minimal starting times for the two trains may also modeled with such a simple choice of $\beta$ (where only one block is initially blocked) together with the same point-set maps (31), (32). This model is useful if one does not know a priori the sequencing of trains on the common track.

A third model may be used if it is (as extra input data) known which train will be ahead (say train 1) and which will follow (say train 2). Then a local but (in $i$) nonsymmetric definition of the point-set maps may be choosen as follows:

$$\texttt{block/unblock}(i,j) \;=\; \begin{cases} \emptyset & i=1,2 \quad j=1, \\ \{(2,j-1)\} & i=1, \quad j=2,\ldots,m_s, \\ \emptyset & i=2, \quad j=2,\ldots,m_s, \end{cases} \tag{34}$$

but in this case initial conditions as in (33) are not useful, because they produce an unrealistic behavior even in the case train 1 is faster than train 2. But if the trains start in the correct sequencing (say train 1 in block $S_2^1$ and train 2 in $S_1^2$) with a consistent initial configuration as

$$\beta_j^i = \begin{cases} 1 & i = 2, j = 1, \\ 0 & \text{otherwise}, \end{cases} \tag{35}$$

then the same simulation behavior is obtained as with the fully symmetric point set maps (31), (32), because in this case the blockings (and unblockings) initated from train 2 (as defined in (31), (32)) will never act as a restriction for the movement of train 1.

Therefore for any realistic behavior one needs the modelling of the local action of the signal system like (31), (32) or (34). But in some situations it is very useful to guarantee some priority rules using an initial configuration of $\beta$ as in the first model which is not completely induced from the blocking map. This will be shown in the next example.

## 8.2 Join of two lines

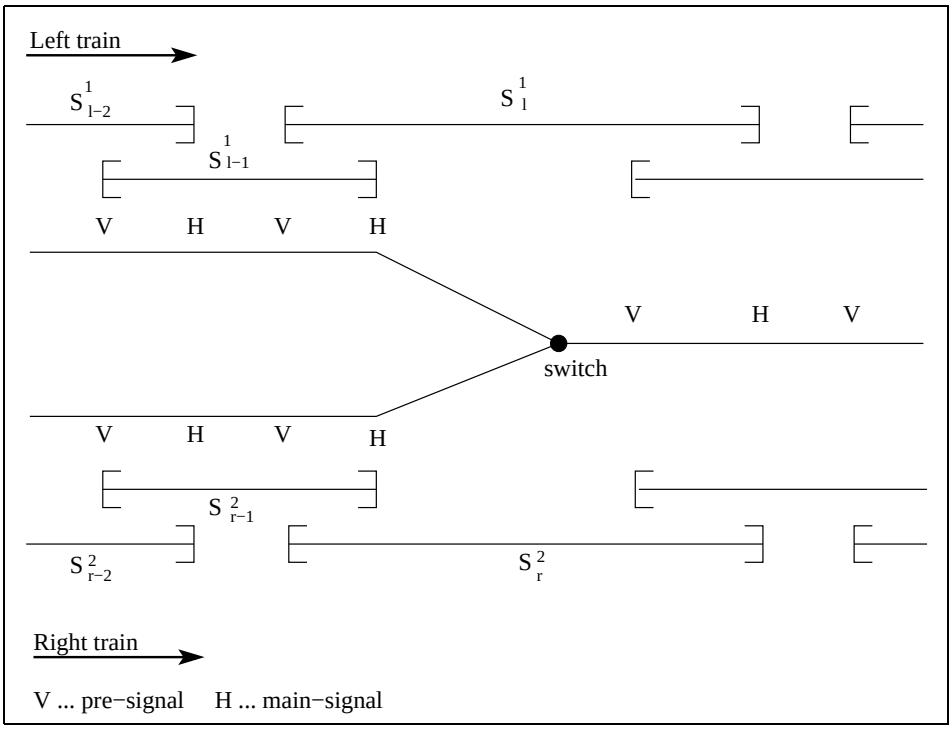


Figure 14: Two lines joining at a switch.

In the situation that two lines join at a switch first we have to prevent from collisions as above. Secondly we can guarantee by an appropriate initial configuration that the trains pass the switch in a given order. As shown in Figure 14 we have a left and a right train going to pass the switch.

We label the relevant blocks of the left train by $\ldots, S_{l-1}^1, S_l^1, S_{l+1}^1, \ldots$ and the ones of the right train by $\ldots, S_{r-1}^2, S_r^2, S_{r+1}^2, \ldots$ The two safety blocks including the switch are $S_l^1$ and $S_r^2$ as depicted.

First we have to model action of the security system at and above the switch. This is done by point-set maps constructed like in the second model of the last example, compare with (31, 32):

$$\texttt{block/unblock}(i,j) \quad = \quad \begin{cases} \emptyset & i=1, \quad j \leq l-1, \\ \{(2, r+(j-l)-1)\} & i=1, \quad j \geq l, \\ \emptyset & i=2, \quad j \leq r-1, \\ \{(1, l+(j-r)-1)\} & i=2, \quad j \geq r. \end{cases} \tag{36}$$

Because the two trains drive independently up to the switch, a consistent initial configuration is $\beta \equiv 0$. This realizes for all choices of minimal starting times a first-come-first-serve strategy at the switch without any systematic priority rules. i.e. the sequencing of the trains on the common track after the switch depends only on the order the trains arrive in blocks $S^1_l$ resp. $S^2_r$.

In most of the situations coming from schedule based train movement, there is a priori given information on the planned or forced sequencing of the trains at the switch and therefore on the following common track. In the simplest case it is derived from the timetable information itself. Now for the common track after the switch a priority rule (say left train first) will be realized with an initial configuration similar to (28):

$$\beta^i_j = \begin{cases} 0 & i=1, \\ 0 & i=2, \quad j \leq r-2, \\ 1 & i=2, \quad j \geq r-1. \end{cases}$$

This models an initial blocking of the common part of the track for the right train/train 2; with identical `block/unblock` mappings as defined in (36) this blocking for train 2 will be never released and the train would never leave block $S^2_{l-1}$. Therefore one has to modify the `unblock` mapping as

$$\texttt{unblock}(i,j) \quad = \quad \begin{cases} \emptyset & i=1, \quad j \leq l-2, \\ \{(2, r-1)\} & i=1, \quad j = l-1, \\ \{(2, r+(j-l)-1), (2, r+(j-l))\} & i=1, \quad j \geq l, \\ \emptyset & i=2, \quad j \leq r-1, \\ \{(1, l+(j-r)-1)\} & i=2, \quad j \geq r, \end{cases}$$

Now there are enough unblocking actions for train 2 after train 1 has passed over the switch.

The table-oriented storage of the point-set maps for this model as generated in a preprocessing step of the program is shown in Tables 1 and 2 for the simple case $r = l$.

Because in this scenario one knows a priori the sequencing of the trains on the common track, some significant simpifications - as in the derivation of (34) - are possible. First the blocking and unblocking actions of train 2 for train 1 may be skipped. In the initial configuration of $\beta$ there is need only for one extra blocking at the switch together with an appropriate unblocking, because keeping the security distance of at least one block between the trains is guaranteed by the incorporated model of the standard security system acting locally. Altogether one gets

$$\beta^i_j \quad = \quad \begin{cases} 1 & i=2, \quad j=r-1, \\ 0 & \text{otherwise}, \end{cases}$$

| block(i,j) | | |
|---|---|---|
| $j \quad \backslash i$ | 1 | 2 |
| $\vdots$ | $\vdots$ | $\vdots$ |
| $l-2 = r-2$ | $\emptyset$ | $\emptyset$ |
| $l-1 = r-1$ | $\emptyset$ | $\emptyset$ |
| $l = r$ | $\{(2, r-1)\}$ | $\{(1, l-1)\}$ |
| $l+1 = r+1$ | $\{(2, r)\}$ | $\{(1, l)\}$ |
| $l+2 = r+2$ | $\{(2, r+1)\}$ | $\{(1, l+1)\}$ |
| $l+3 = r+3$ | $\{(2, r+2)\}$ | $\{(1, l+2)\}$ |
| $l+4 = r+4$ | $\{(2, r+3)\}$ | $\{(1, l+3)\}$ |
| $\vdots$ | $\vdots$ | $\vdots$ |

Table 1: Storage of point-set map `block` for two joining lines

| unblock(i,j) | | |
|---|---|---|
| $j \quad \backslash i$ | 1 | 2 |
| $\vdots$ | $\vdots$ | $\vdots$ |
| $l-2 = r-2$ | $\emptyset$ | $\emptyset$ |
| $l-1 = r-1$ | $\{(2, r-1)\}$ | $\emptyset$ |
| $l = r$ | $\{(2, r-1), (2, r)\}$ | $\{(1, l-1)\}$ |
| $l+1 = r+1$ | $\{(2, r), (2, r+1)\}$ | $\{(1, l)\}$ |
| $l+2 = r+2$ | $\{(2, r+1), (2, r+2)\}$ | $\{(1, l+1)\}$ |
| $l+3 = r+3$ | $\{(2, r+2), (2, r+3)\}$ | $\{(1, l+2)\}$ |
| $l+4 = r+4$ | $\{(2, r+3), (2, r+4)\}$ | $\{(1, l+3)\}$ |
| $\vdots$ | $\vdots$ | $\vdots$ |

Table 2: Storage of point-set map `unblock` for two joining lines

$$\texttt{block}(i,j) = \begin{cases} \emptyset & i=1, \quad j \le l-1, \\ \{(2, r+(j-l)-1)\} & i=1, \quad j \ge l, \\ \emptyset & i=2, \end{cases}$$

$$\texttt{unblock}(i,j) = \begin{cases} \emptyset & i=1, \quad j \le l-2, \\ \{(2, r-1)\} & i=1, \quad j = l-1, \\ \{(2, r+(j-l)-1)\} & i=1, \quad j \ge l, \\ \emptyset & i=2, \end{cases}$$

which provides for all minimal starting times exactly the same simulation behavior as the previously derived complex model having about three times more non-trivial entries in the point-set maps.

### 8.2.1 Some calculations for this example

The switch is located at kilometer 13.5, i.e., after this point both trains use the same path. We did 4 calculations with the following properties:

1. There is no train scheduling. Train 1 starts at time 0 and train 2 after 90 seconds.
2. There is no train scheduling. Train 1 starts after 90 seconds and train 2 at time 0.
3. There is a train scheduling, enforcing that train 1 can pass the switch first. Train 1 starts at time 0 and train 2 after 90 seconds.
4. There is a train scheduling, enforcing that train 1 can pass the switch first. Train 1 starts after 90 seconds and train 2 at time 0.

This gives us the following results:

Test 1 : Train 1 arrives at first
Test 2 : Train 2 arrives at first
Test 3 : Train 1 arrives at first
Test 4 : Train 1 arrives at first

The path-time-diagrams are shown in the Figures 15,16,17 and 18.

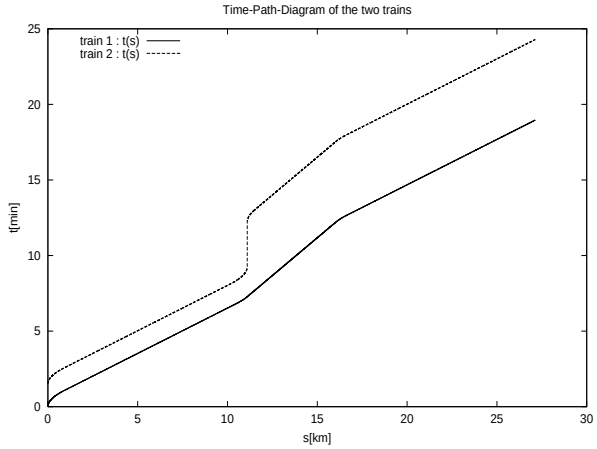


Figure 15: Test 1

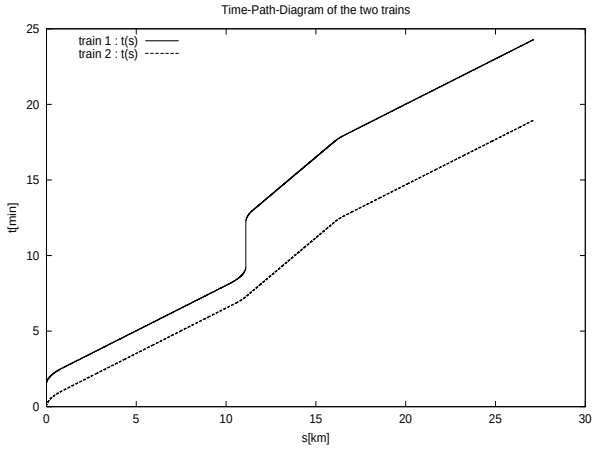


Figure 16: Test 2

# 9 The equations of motions for several trains

In this section we discuss the equations of motion for several trains interacting via the safety concept. Each train moves in the same fashion via the equations of motion given by (10) and the time optimal control problem for every train individually is to minimize $t_f^i - t_0^i$ subject to

$$\begin{aligned} \dot{x}^i(t) &= Ax^i(t) + B^i[u_f^i(x^i) + u_v^i(t)], \\ x^i(t_0) &= \begin{bmatrix} s_0^i \\ v_0^i \end{bmatrix}, \end{aligned}$$

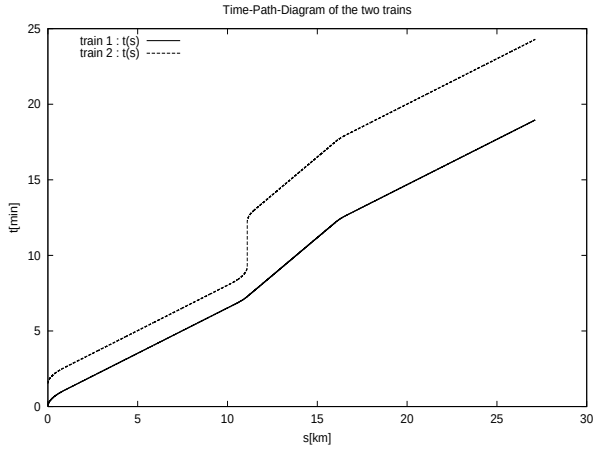


Figure 17: Test 3

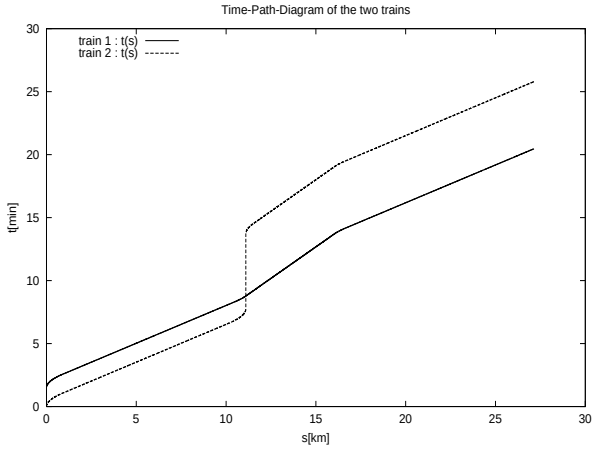


Figure 18: Test 4

$$\begin{aligned} x^i(t_f) &= \begin{bmatrix} s_f^i \\ \cdot \end{bmatrix}, \\ u_v^i(t) &\in [u_{min}^i, u_{max}^i(v^i(t))]. \end{aligned}$$

Here the matrix $A$ is the same for all trains, but the matrix $B$ is different, since it contains the mass of the train, i.e.,

$$\begin{aligned} A &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \\ B^i &= \begin{bmatrix} 0 \\ \frac{1}{m^i} \end{bmatrix}, \qquad i = 1, \ldots, n_z. \end{aligned}$$

For a single train we just had velocity constraints given by

$$v(s(t)) \in [0, v_{max}(s(t))] \quad \forall t,$$

and analogously we have for several trains

$$v^i(s^i(t)) \in [0, v_{max}^i(s^i(t))], \quad \forall t, \quad i = 1, \ldots, n_z.$$

On the basis of the safety concept we had derived the braking curves $b_j(s)$ (23) in the different safety blocks and the indicator functions $\chi_j(s^1, \ldots, s^{n_z})$ (24), that activate or deactivate the breaking curves and thus we generate the relevant velocity constraints $v_{rel}^i(t, s^i)$ in (27).

Since the indicator functions $\chi_j$ depend on the current position of all interacting trains (which are again time dependent), we now have position- and time-dependent constraints and we obtain a coupled system

$$\begin{aligned} \dot{x}^i(t) &= Ax^i(t) + B^i[u_f^i(x^i) + u_v^i(t)], \\ x^i(t_0) &= \begin{bmatrix} s_0^i \\ v_0^i \end{bmatrix}, \end{aligned}$$

$$
\begin{aligned}
x^i(t_{end}) &= \begin{bmatrix} s^i_{end} \\ . \end{bmatrix}, \\
u^i_v(t) &\in [u^i_{min}, u^i_{max}(x^i_1(t))], \\
v^i(t) &\in [0, v^i_{rel}(t, x^1_1(t), ..., x^{n_z}_1(t))],
\end{aligned}
$$

where the state constraints determine the variable part of the control $u^i(t)$.

There are now several options for time optimality, since every train has an individual $t^i_f$. Consider the vector of travel times

$$
T_t = \begin{bmatrix} t^0_f - t^0_0 \\ t^1_f - t^1_0 \\ \vdots \\ t^{n_z}_f - t^{n_z}_0 \end{bmatrix}.
$$

By choosing a norm in different ways we can optimize different criteria, but we may also use a componentwise optimization to minimize all individual travel times under constraints of priority of one train over the other, i.e., after we have fixed the order of trains at common pieces of the network, then for every train we have an individual optimization.

Here we follow the latter strategy, i.e., in any case of conflict between trains we fix the order of the trains a priori, and then use the strategy to determine the time optimal case for every train individually. In this way we can use the strategy developed for one train with extra constraints. This models the individual behavior of all drivers simultanously trying to drive their trains time optimal. On the mathematical side the advantage of this approach is that we do not have to solve a boundary value problem but, as in the case of one train, we can determine the minimal travel time as follows.

In the case that there are no state constraints, it is well-known [5] that the optimal control is a bang-bang control, i.e., the control switches between its minimal and maximal value. In our case we also have state constraints. This leads to the following time-optimal control strategy. If the train is restricted by a braking curve then, since we have assumed that the braking curve can be realized, the control is fixed, since we have to guarantee the safety constraints.

If, due to unblocking during the braking period, the braking curve is deactivated then the strategy is changed to the same situation as in the case of one train, where we only have train-independent velocity constraints.

In such a situation we must set the variable part of the control in such a way that the maximal allowed velocity is kept and that the exterior forces are compensated, i.e., we have

$$
\begin{aligned}
\dot{s}^i(t) &= v^i(t), \\
v^i(t) &= v^i_{rel}(t, s^i(t))
\end{aligned}
$$

for all $t$, in which we stay on the constraint, where $v^i_{rel}$ is as in (27). This allows to determine the travel time in the same way as for one train.

**Remark 1** It should be noted that due to the interaction of trains and the priority rules that are fixed a priori, the time optimal strategy that we have described in this section in general does not lead to a unique control strategy. The optimal vector of minimal travel times $T_t$ is unique but there is quite a lot of freedom in the choice of control. This can be easily seen in

the case of a fast train following a slow train on one single track, if this situation is allowed to happen due to the given priorities. Here it is clear that the second train has many options for the indivdual time-optimal control. It can always keep the minimal possible distance to the first train or it can just wait at some point until it can use the whole line without braking curves. In order to make the optimal control unique, in such a situation further optimality criterias have to be prescribed.

# 10 Realization of the simulation program

In this section we describe some details on the implementation (in C++) of the simulation method that realizes a time optimal control. The simulation runs until all trains have arrived in their final destination point.

## 10.1 Preprocessing

In a preprocessing step (as in the case of one train) we determine the controls that have to be applied in order to guarantee that always the maximal velocity in every block is kept and also we determine all possible braking curves that are necessary to guarantee the safety concepts if an interval is blocked.

As data structure we use a vector in which the different trains are stored as (C++) objects `train ride`. Each of this objects contains all the data for one train and the path that it is using as well as the necessary methods for the simulation. The data for the path are given by a doubly connected list of subintervals. Furthermore, there are objects that administrate the safety blocks and that realize the blocking and unblocking of blocks.

Before the simulation the tables for the point-set maps `block` and `unblock` are checked for consistency. To determine the braking curves that are needed to guarantee that in the next interval we meet the velocity constraint, we proceed backwards as in the case of one train, see Section 4.

For the braking curves that guarantee the safety concept we first introduce possible dummy blocks that guarantee that every part of the path has exactly two safety blocks and then we assign labels `siblo1` and `siblo2` corresponding to the functions $\sigma^i$ (26). Then we order the local labels so that a change of the value of these functions corresponds in a change of the blocking and unblocking assignment of the safety blocks. This is necessary because it is possible that in the data file the labels of the safety blocks could be changed in the $i$-th subinterval of a path compared to the $(i+1)$-st subinterval. But this swapping of the numbers is not a real change of a safety-block. It is also possible that only one safety block changes but the other one is stored in the next path interval on `siblo2` rather than on `siblo1`.

The following algorithm guarantees that safety blocks are stored correctly and that a change in the label corresponds to a change in the safety block.

For this let $i$ be an index to label the subinterval, i.e., an element of the list.

1. Start at the beginning of the list of blocks with $i := 0$.
2. `number1`:=`siblo1`(i)
   `number2`:=`siblo2`(i)
3. Increment $i$
4. If `number1`=`siblo2`(i) then exchange `siblo1`(i) and `siblo2`(i)

5. `number1:=siblo1(i)`
   `number2:=siblo2(i)`
6. GOTO 3 until the whole list has been checked.

Then we go backwards through the list and determine the maximal starting velocity at the beginning of each block that guarantees the stopping at the end of the safety block. If at the end of a safety block the maximal velocity is 0, then we determine the velocity at the beginning of the block and use this as initial velocity for the next part of the path until the safety block changes. Then the maximal velocity is zero again. In this way we obtain for the whole path the maximal velocity associated with the braking curves $b_j(s)$.

## 10.2 The forward simulation

In this section we describe the simulation procedure.

To initialize the simulation we set the variable that counts the number of trains that have not reached their end-point to $n_z$. Then in a round-robin fashion, we do an integration step of the simulation for every train until all trains have reached their end-points. In this process we face several difficulties. One of the major problems from the numerical analysis point of view is that the time stepping procedure has to adapt to the wide variety of different length of the safety blocks. It is currently under investigation what is the most efficient way to do this, since due to the safety blocks all the trains have to be synchronized. Furthermore, typically the current step sizes do not get us exactly to the ends of blocks. For this reson we currently work with a constant stepsize in time and if we pass with such a time-step over boundary points of blocks, then we interpolate the values to the boundary points, if necessary at once or possibly also in a post processing procedure.

It is currently under investigation how to efficiently make use of variable stepsize integration methods.

In an integration step we always test first, whether the train for which we are about to do an integration step has reached its final destination. If not then we proceed with the next train. Then we test whether the train is already allowed to proceed, or whether it still has to wait. This happens if the simulation time is before the starting time or when due to a stop in a station the train has to keep waiting.

In all other cases we proceed and perform the next time step as follows.

- If a safety block changes then we test whether we can leave the block. If so, then we call the method for blocking and unblocking of the safety blocks. Then either this train is allowed to leave the block or we set its position to the end of the block and the velocity to zero and wait.

- If the time step reaches the final destination or beyond, then we interpolate the values for time and velocity, note that the train has arrived in the final destination and decrement the number of trains which have not arrived yet.

- If we have entered a stopping interval, then we interpolate the arrival time and set the waiting time as the sum of arrival and stopping time.

### 10.3 Data Interface

The infrastructure, the specific data of each train as well as the safety concept enter the program via ASCII data-files.

In the *safety block file* the number $m_s$ of safety blocks, the initialization of the vector $\beta$ as well as the tables of point-set-maps `block`$(j)$ und `unblock`$(j)$ are stored.

In the path file of a train, the starting time and velocity of the train, the number of blocks, their name and their length, the slope, the maximal velocity, the maximal braking power, the stopping times as well as the global labels of the two local safety blocks are stored. As an example consider the following file

```
#
#  Example of a path file
#
#  path file for train A
#
#
# Starting time in seconds after $0$
 0.0
# Starting velocity in km/h
 0.0
#
#  number of path intervals
  9
# data of path intervals
#
# length slope   vmax    amin    stopping-time  safety blocks       name
# [m]           [km/h] [m/s^2]     [s]
  5000.0  0.0    100.0   -0.4      0.0           1    2            int_1
  3000.0  0.08   100.0   -0.4      0.0           0    2            int_2
  3500.0 -0.03   100.0   -0.4      0.0           2    3            int_3
     0.0  0.0      0.0   -0.4    600.0           2    3            station_xy
  1000.0  0.03    80.0   -0.4      0.0           0    3            int_4
   200.0  0.0     30.0   -0.4      0.0           0    3            int_5
  2000.0  0.03    80.0   -0.4      0.0           3    4            int_6
  2000.0 -0.01    60.0   -0.4      0.0           0    4            int_7
  1000.0  0.0    100.0   -0.4      0.0           4    5            int_8
# END
```

The *train data file* contains the data relevant for the specific train, the mass in kg including an extra mass to compensate for the rotation, the coefficients for the approximation of the friction forces according to

$$u_{fric}(v) = k_0 + k_1 v + k_2 v^2$$

and a sampling of the velocity dependend maximal acceleration force $u_{max}(v)$ at $v = 0, 0.1, 0.2 \cdots$ m/s measured in Newton.

```
#
# Example of train data file
#
# Mass in kg
  876000
# friction coefficients  k_0,k_1,k_2
  1000.0 0.0 10.3
# Approximation umax(v)
#       v[m/s]       umax[N]
        0.000000     400137.000000
        0.100000     400029.000000
        0.200000     399921.000000
        0.300000     399813.000000
        0.400000     399704.000000
        0.500000     399596.000000
        0.600000     399488.000000
        0.700000     399380.000000
        0.800000     399272.000000
        ...
```

The safety block files are used for defining the safety-conditions between trains and to set the sequence of different trains.

This files consist of lines in the form

```
#
#  comments
#
  1 BLOCK 3
  4 UNBLOCK 5
  6 INITIAL
  ...
```

This means that the safety-block number 1 blocks the safety-block number 3, block 4 unblocks number 5 and number 6 has one initial-blocking. Repeating of INITIAL -Lines with the same block-number will increase the number of blockings on the safety-block. It is useful to distinguish safety-conditions (pairs of lines with `BLOCK` and `UNBLOCK`) from precedence-conditions (pairs of lines with `UNBLOCK` and `INITIAL`) and put the information in different files. To concatenate all this files for the FZR2D-program there exists the PERL-Script `generate-siblo.pl` .

The *parameter file* is used for the control of the simulation program. This is an ASCII-File with lines of the form.

```
parameter_name = value
```

For parameters that are not assigned, default values are used. The order of parameters is arbitrary, if multiple assigments are made always the last is used.

**plot_para** Dumping the parameters (default=1)

**plot_wegliste_vor_glaetten** Printing the path-list before correction of $vmax(s)$ (default=1).

**plot_wegliste_nach_glaetten** Printing the path-list after correction of $vmax(s)$, i.e., with the extra inserted jump-off intervals and braking-curves (default=1).

**plot_vmax_vor_glaetten** Generating a Gnuplot-datafile for plotting the maximal velocity (default=1).

**plot_vmax_nach_glaetten** Generating a Gnuplot-datafile for plotting the corrected maximal velocity (default=1).

**plot_bremskurve_aller** Control-parameter for the generation of datapoints for plotting braking-curves. We generate a data-point all `plot_bremskurve_aller` meter. (default=1.0 m).

**plot_bremskurven** Generating a Gnuplot-datafile for plotting all braking-curves (default=1).

**plot_vmax_rel** Generating a Gnuplot-datafile for plotting the minimum of all restrictions (default=1).

**v_inf** Value of the maximal velocity for Dummy-blocks. It should be a value , which is never reached. (default=500 km/h)

**simu_stepsize** Time-stepsize of the simulation (default=1.0 s)

**simu_maximal** Maximum of simulation time. The program will brake with an exception if it is reached. (default=86400 s, this are 24 hours )

# 11 Some test cases

## 11.1 The analytical test case

In this section we give some tables with the analytical test case from Chapter 6.3 for one train.

They show the dependence of the correctness and the degree of the interpolation, see Tables (3) to (8). An interpolation is needed to compute correct values for time and velocity at the endpoints of the intervals, because the ode-solver only produce data at gridpoints.

The columns of the tables describe the following values:

**degree** Degree of the interpolation polynom

**eps_v** Parameter of the step-size-control

**time** Computed time

**error in time** Error in the computed time (compared with the analytical solution)

**velocity** Computed terminal velocity

**error in velocity** Error in the computed terminal velocity (compared with the analytical solution)

### 11.1.1 Variant to reach the end of the interval

We depict tables for different polynomial degrees in the interpolation at the endpoint.

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

Table 3: Error in case interval end with polynomial degree 1

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

Table 4: Error in case interval end with polynomial degree 2

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

Table 5: Error in case interval end with polynomial degree 3

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

Table 6: Error in case interval end with polynomial degree 4

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

Table 7: Error in case interval end with polynomial degree 5

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

Table 8: Error in case interval end with polynomial degree 6

## 11.2 The test case 22127-1013

The following example was computed on an Intel Pentium 2, 450 MHz with 384 MB RAM. It consists of 80 Kilometers with a subdivision of 1275 intervals to compute. The following computing-times were needed in the different parts:

| Creating the list | Computinf switching-off-points | Driving the route | Postprocessing |
|---|---|---|---|
| 0.04 s | 0.01 s | 0.24 s | <0.01 s |

We had a total computing time of 0.29 seconds (without reading ASCII-data)

In the following figures the maximal velocity, corrected velocity, v(s), t(s), outer forces, actuation force and braking acceleration are depicted.

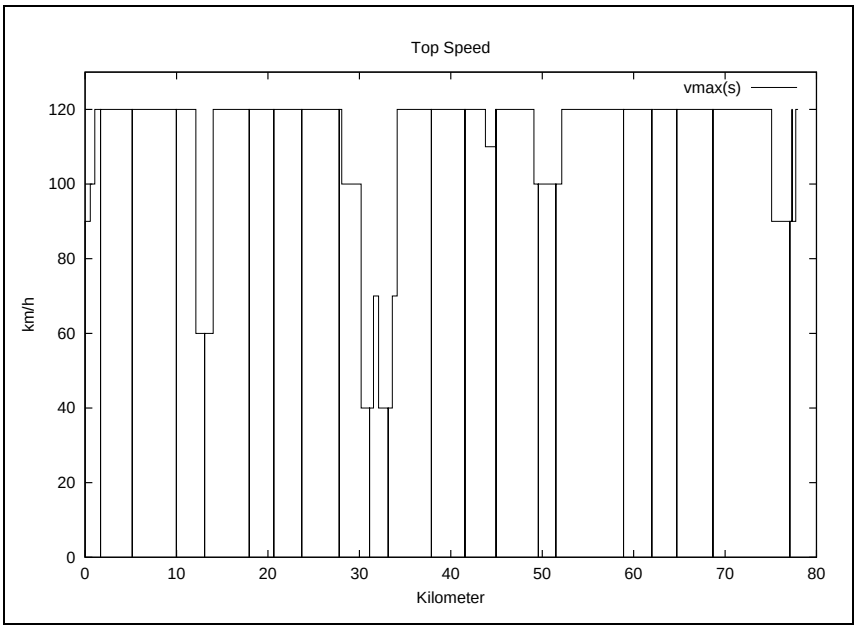


Figure 19: Case 22127-1013: Top Speed

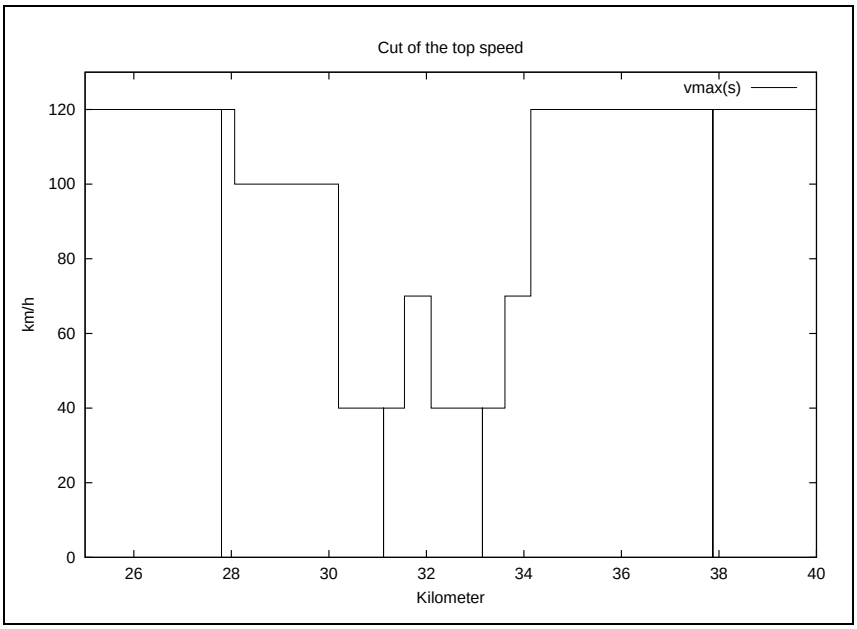


Figure 20: Case 22127-1013: Cut of Top Speed

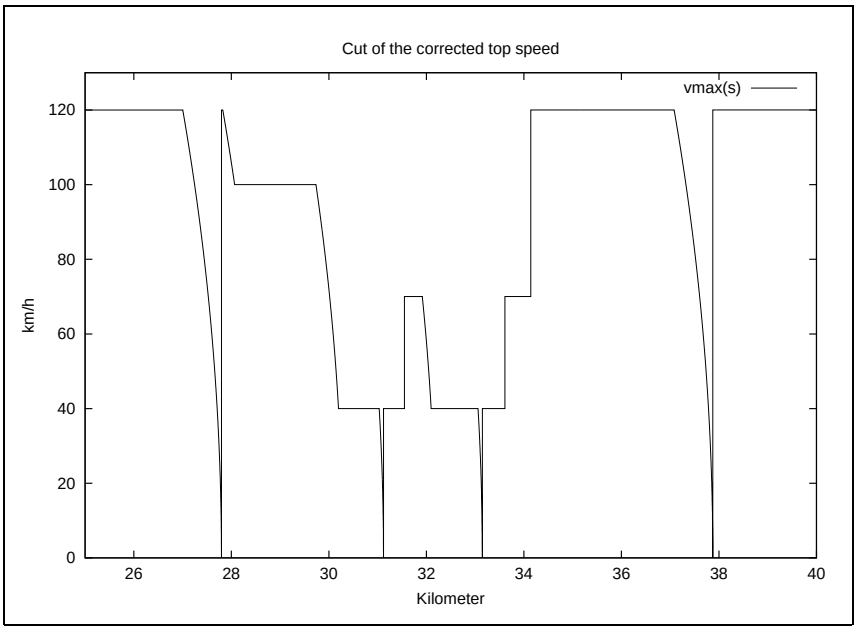


Figure 21: Case 22127-1013: Corrected velocity

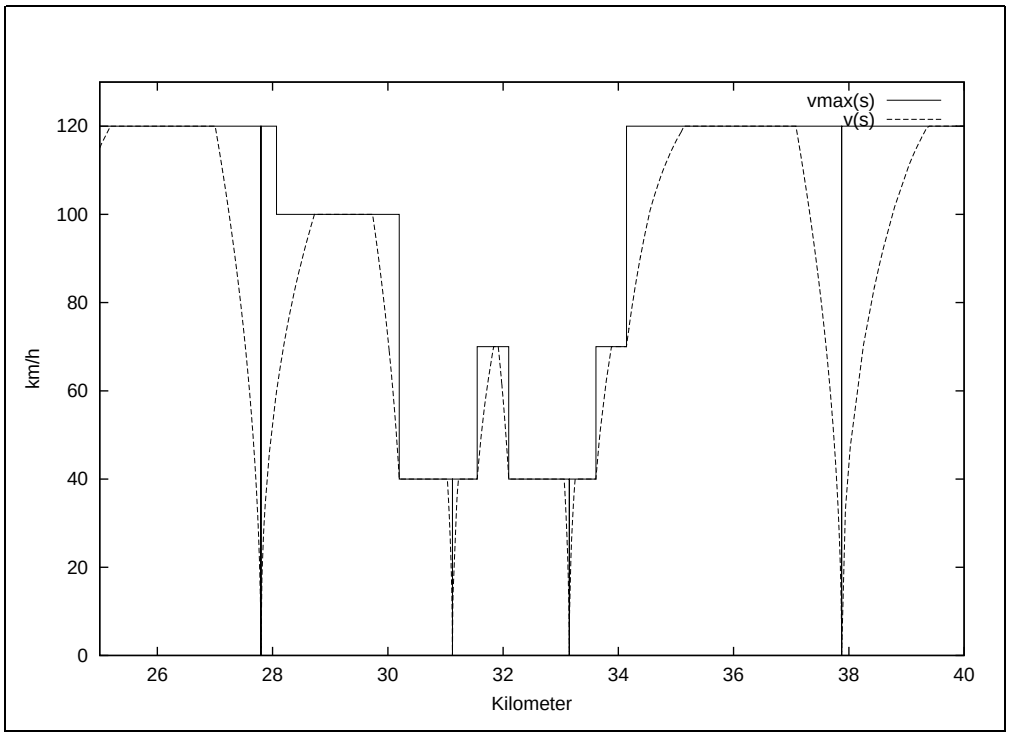


Figure 22: Case 22127-1013: velocity - path - diagram

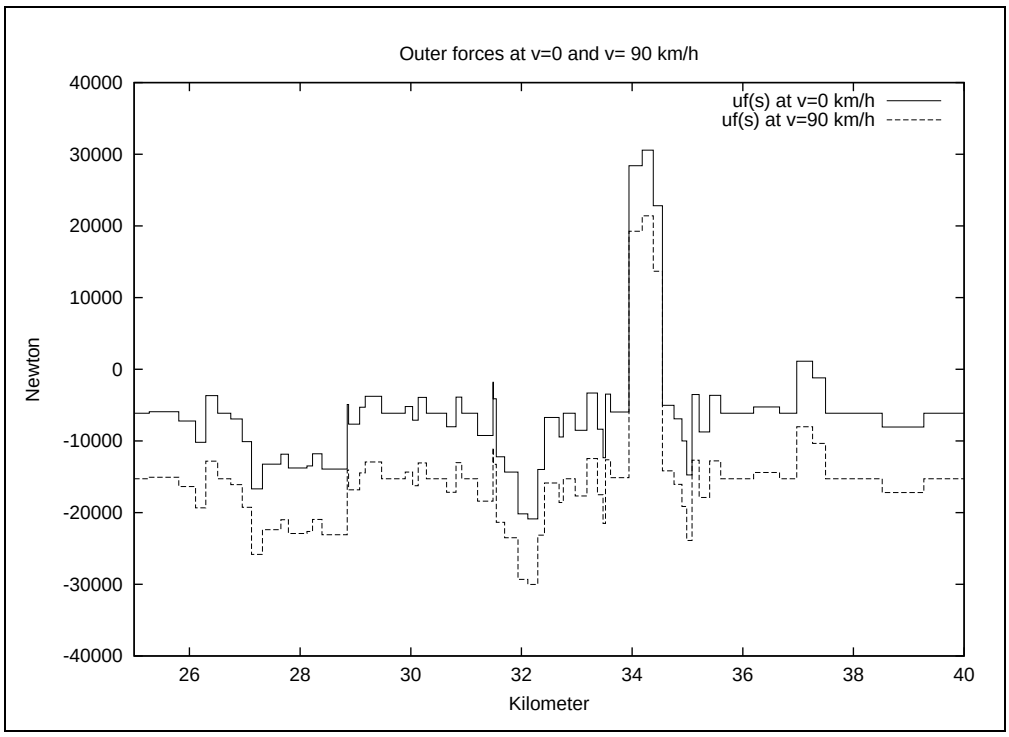


Figure 23: Case 22127-1013: Outer forces

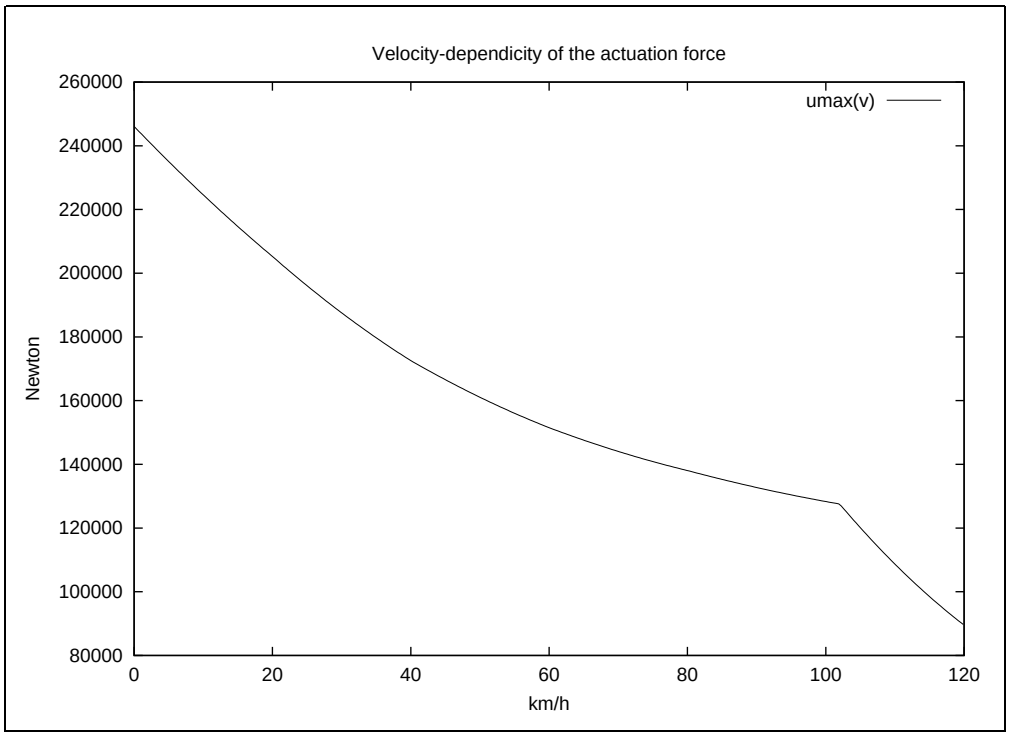


Figure 24: Case 22127-1013: Actuation force

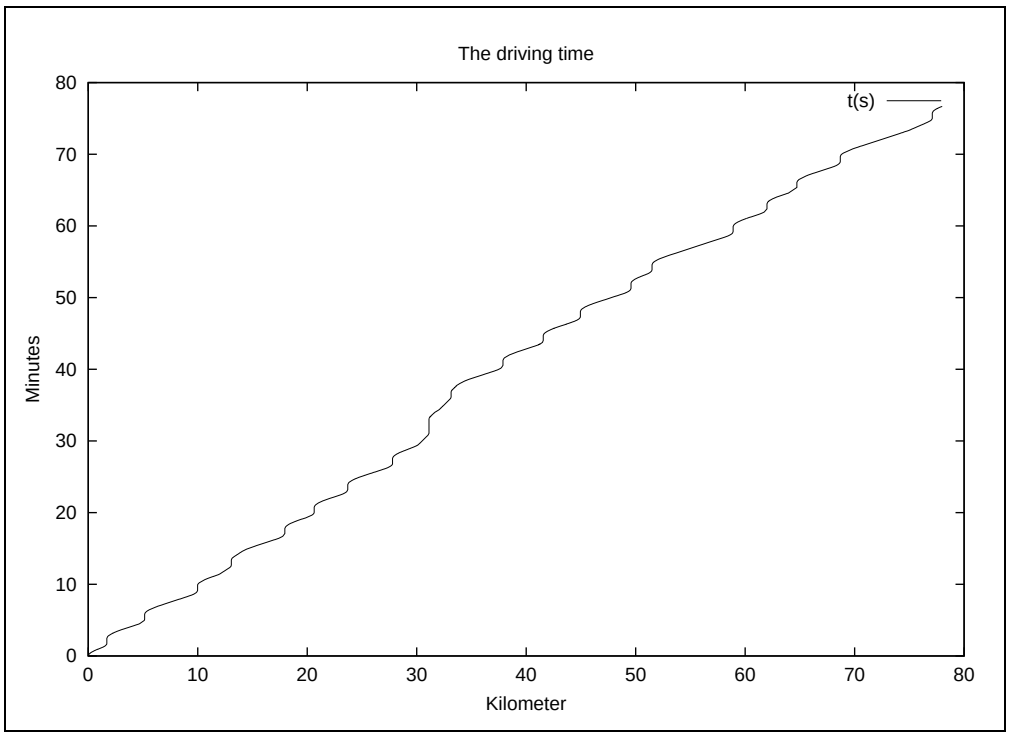


Figure 25: Case 22127-1013: time - path - diagram

# 12 Conclusion

We have discussed the mathematical modelling and simulation of schedule based rail traffic including a realistic safety concept. On the basis of time optimal control efficient numerical methods have been developed that allow the joint simulation of many interacting trains.

# References


[1] U. Ascher, R.M.M. Mattheij, R.D. Russel. *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*. SIAM, Philadelphia 1995.

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

[3] Ch. Grossmann, J. Terno. *Numerik der Optimierung*. Teubner, Stuttgart, 1997.

[4] H. Herold. *C-Kompaktreferenz*. Addison-Wesley, Bonn, 1999.

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

[6] J. Pachl. *Systemtechnik des Schienenverkehrs*. Teubner, Stuttgart; Leipzig, 1999.

[7] R. Schaback, H. Werner. *Numerische Mathematik*. Springer-Verlag, Berlin, Heidelberg, New York, Stuttgart, 1992.

[8] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Springer, 1993.

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


# A On the programs

Both programs FZR1D and FZR2D are written in ISO-C++ and tested with Gnu-Compiler gcc 2.95.2 . To build the programs one must unpack the archives, go to the root-directory and type make. If the compiler is not a gcc one will have to change the `CCOM` Variables in the Makefiles.

Details for handling of the programs and compiling different versions also described in the readme and make files of the packages. The most important switches of both programs are described in the following two sections.

## A.1 FZR1D

Different versions of FZR1D can be compiled by setting some compiler-macros in the Makefile. Setting `FZR_TRACE_LOGIK` will produce many output in the program. Setting `ZEITMESSUNG` is useful to stop the computation time.

## A.2 FZR2D

In the same way as in the 1D case one can set different macros to get different versions of the program. To get a program with many debugging-output set `FZR2D_DEBUG`. Setting `FZR2D_SIMUSTEPS` makes the program output the steps of the ODE-solver, and setting `FZR2D_ZEITMESSUNG` will give a measurement of computing-time.