A Fast Algorithm for Computing the Running-Time of Trains by Infinitesimal Calculus

Thomas Schank, Ph.D.
e-mail: Dr.Th.Schank@gmail.com

Abstract
We develop a fast algorithm for computing the running time (as well as energy consumption) of trains based on solving the underlying differential equation exactly, rather than applying any of the conventional approximation techniques. We show that an implementation of our algorithm outperforms those traditional methods by a factor of several decades in speed. We thus conclude that our contribution could boost optimizing efforts significantly.

Keywords
running time of trains, time table planning, optimization, algorithms, implementation, efficiency, energy conservation, differential equations, calculus, energy consumption

1 Introduction

1.1 Motivation and Context

Timetable planing is one of the most important tasks in operating trains and the computation of the running times itself is one of the foundations of timetable planning. The mechanical fundamentals for computing the running times of trains has been studied early, e.g. by Strahl in 1913 [18] and Davis in 1926 [6]. It is also subject of textbooks in the field, for example Wende in 2003 [20]. In the majority the target of study or description is the application of various methods with respect to gaining an acceptable approximative result. In this work we focus on the efficiency in practice of gaining an exact result for a certain format of input. Fast computation of the running time can boost applications in particular with respect of optimization and online computing.

Timetable planning has been very well studied from the perspective of scheduling and networks, see e.g. [11, 15, 17]. These are based on the presumption of a static traversal time between two nodes in the network. This is to some extend an applicable assumption for dedicated high speed lines that do not share tracks. If however, the track is shared between trains of different speeds, or even there is only one track for both directions, then the trains influence each other with respect of their running-times which has to be taken into account during computation. It requires frequent reevaluation in particular in the context of automatic optimization.

Fast computation of running time also plays a role in online algorithms that adapt to discrepancies between the planed and the actual disturbed schedule. Again optimization for mutually dependent target functions are of concern. Efforts for adaptive conservation of energy are based on the same foundations.

1.2 Organization of this Work

The reminder of this work is organized as follows: we will derive a common mathematical description of the various forces that act on a composition during it's motion in Section 2. In particular we show that a certain formulation of the propulsion-force has the same mathematical description as the combined friction-forces (Section 2.3). Further we show, that the whole dynamics is represented by exactly one differential equation (Section 2.4). In Section 3 we give a solution to the derived differential equation and argue that all solutions are equivalent using well known theorems. From the solution we derive all necessary equations required by an algorithm for computing the running time. We layout such a generic algorithm in Section 4. In Section 5 we discuss our implementation of the algorithm in conjunction with the given equations. We also show by an experiment that the algorithm based on the exact solution of the equation outperforms traditional approximative methods in execution time. We conclude with Section 6.

2 The Differential Equation of the Dynamics of Trains

Various forces act on a composition during it's motion. Resistance forces, given by air drag, mechanical friction and mechanical deformation act against the direction of motion by dissipating unrecoverable energy to the environment. We discuss the model in Section 2.2. The engine supplies forces to accelerate or to hold a given speed which act in the direction of motion. In some cases this force can be applied in the opposite direction for deceleration aimed at regaining some part of the dispensed energy for propulsion. A method for modeling engine forces is given in Section 2.3. We start with the simplest force: the one given by climbing or descending a slope.

2.1 Force given by Ascending or Descending

Potential energy is exchanged with the gravitational field of the earth if the track is not flat. The corresponding force is computed with

$$F_H = m\,g\sin\varphi = h_1 \tag{1}$$

where φ is the angle perpendicular with respect to the surface of the earth (i.e. perpendicular to the gravitational field). Note that the force is independent of the velocity and in particular constant for a segment of a track with $\varphi = \text{const}$.

2.2 The combined Formula for Resistance Forces

The various resistance forces can be modeled together in an equation as a second degree polynomial with respect to the speed v:

$$F_R(v) = r_0 + r_1 v + r_2 v^2. \tag{2}$$

Equation 2 includes the drag from the motion in the air, mechanical friction in the engine and coupling as well as between rail and wheels, and finally those given by mechanical deformation. This equation has been used since the age of steam locomotives as given by Strahl in 1913 [18] and early electric driven locomotives given by Davis in 1926 [6]. In

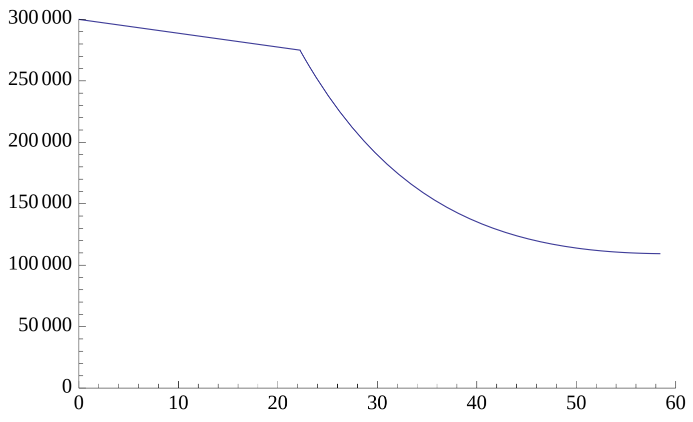


Figure 1: Propulsion-Force F_e in Dependency of v

following years the model has been refined and the parameters of Equation 2 have been taken apart and explained by various independent mechanical characteristics.

Rather modern approaches rely on empiric results that are in turn mapped to Equation 2 by statistical fittings, see e.g. [13]. Note that the coefficients r_1, r_2, and r_2 are not constants in general. However, they are fixed for a particular composition of vehicles. Additionally tracks are in general split in sections for which they can be regarded as such.

2.3 A Second Order Approximation for Engine Forces

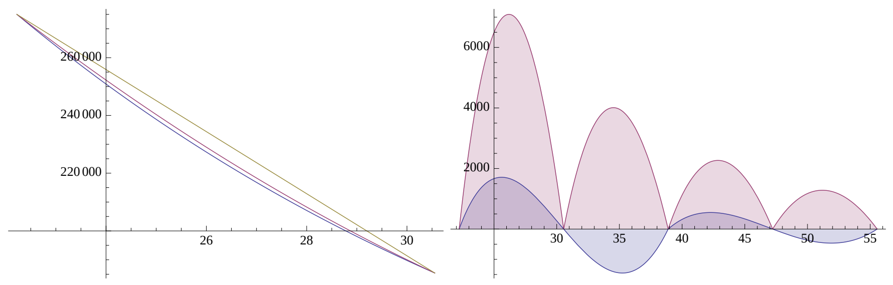
Propulsion-forces of tractive vehicles are usually available in implicit form as diagrams as shown in Figure 1. To make the following arguments reproducible in a simple way we assume that we are given a pull-force F_e explicitly as in Equation 3:

$$F_e[\mathrm{N}](v[m/s]) = \begin{cases} 3.\cdot 10^5 - 1.13\cdot 10^3 v & 0 \le v \le \frac{200}{9} \\ 1.72\cdot 10^4 + 1.05\cdot 10^6 e^{-6.84\cdot 10^{-2} v} + 1.25\cdot 10^3 v & \frac{200}{9} < v \le \frac{550}{9} \end{cases} \tag{3}$$

The force shown in Figure 1 is a subset of an existing pull force (the original extends to higher speeds which are not relevant for the examples shown in Section 5). The linear segment is due to limits of transferring forces to the rail and it is suggested to use a general exponential form to model the curved part given by engines [20].

The diagrams are converted into some form of suitable and explicit mathematical representation. The obvious solution is to use linear approximation. This is suggested in [20] and as a table of x/y values also intended to be used for the *rolling-stock* subschema of *railml* [19]. The yellow line in Figure 2(a) shows the linear approximation of F_e (blue line) from 200/9 [m/s] to 275/9 [m/s] with one segment.

Now, going further one level means to approximate with a quadratic form or a second order polynomial. This isn't actually new either. The most modern rolling-stock database used with the SBB stores the corresponding propulsion-forces in second order polynomial segments. The reason is that the user needs to input much fewer reference points to gain



(a) F_e(blue), F_z (red), F_l (yello)

(b) $F_z - F_e$ (blue), $F_l - F_e$ (red)

Figure 2: Propulsion-Force Detail and Difference

same quality as compared to linear segments. Our motivation is rather to use the same (maximal) order as in Equation 2 which is, by chance, also quadratic.

Equation 4 shows a second order approximation F_z of F_e where the exponential part is represented by two quadratic polynomials (from 200/9 m/s to 350/9 m/s, and from 350/9 m/s and higher). The first part where v is less than 200/9 m/s is left as linear segment (which can be regarded as a special form of a quadratic segment where the coefficient of highest order is zero).

$$F_z[\mathrm{N}](v[m/s]) = \begin{cases} 3.000 \cdot 10^5 - 1.125 \cdot 10^3 v & 0 \le v < 200/9 \\ 7.263 \cdot 10^5 - 2.726 \cdot 10^4 v + 3.128 \cdot 10^2 v^2 & 200/9 \le v < 350/9 \\ 4.237 \cdot 10^5 - 1.120 \cdot 10^4 v + 1.000 \cdot 10^2 v^2 & 350/9 \le v \end{cases} \tag{4}$$

If we would have added F_z in Figure 3 the two curves would appear almost indistinguishable. Therefore a detail is shown in Figure 2(a). The difference of quality is more clearly visible in Figure 2(b) where differences are shown.

2.4 The Differential Equation of Train Dynamics

If we add F_H from Equation 1, F_z from Equation 4 and F_r from Equation 2 we get the full description of the forces that act on the corresponding composition during acceleration. Deceleration can be formulated in the same way and hence the final equation for a segment of speed for v with $v_i \le v \le v_{i+1}$ and a segment of track with constant parameters is

$$F = c_0 + c_1 v + c_2 v^2. \tag{5}$$

If we apply Newtons-Law

$$F = am \tag{6}$$

we finally get the differential equation of dynamics

$$\dot{v} = \alpha + \beta v + \gamma v^2 \tag{7}$$

where $\dot{v} = \frac{\mathrm{d}}{\mathrm{d}t} v = a$ and the mass m is now contained in the parameters α, β and γ. We will now discuss how Equation 7 can be used to compute the motion of trains.

3 Solving the Differential Equation and using the Solution

3.1 A Solution for the Differential Equation

We give now a solution for Equation 7, discuss it, and bing it into context with literature and previous work. Let

$$\kappa = \beta^2 - 4\alpha\gamma, \tag{8}$$

then for $\gamma \neq 0$ (these restrictions are to be discussed, see Section 3.3) and $\kappa \neq 0$ Equation 9

$$v(t) = \frac{-\beta + \sqrt{-\kappa}\tan\left(\frac{1}{2}\sqrt{-\kappa}(t+T)\right)}{2\gamma} \tag{9}$$

with $T \in \mathbb{R}$ a constant, is a solution for Equation 7. This can be verified by applying the operator $\frac{d}{dt}$ on 9.

Equation 9 in Context and Previous Work

In a more general form, where the parameters α, β and γ are functions of t, Equation 7 is known as the *Riccati-Equation* which has no general solution, see e.g. [8, 23]. Obviously the restricted variant has a solution and useful predicates about it can be derived, too. The Picard–Lindelöf [12] theorem provides powerful *existence* and *uniqueness* properties for a ordinary differential equation with given boundary conditions, see also [8]. This guarantees, when applied to our case, that there exists a unique solution which we have found. In particular this also means that solutions that might appear to be different are in fact equivalent.

General methods how to solve ordinary differential equations (to which Equation 7 belongs) are described in textbooks such as [8, 3, 23]. In the context of train dynamics such a procedure for Equation 7 is carried out in Wende [20], albeit with a somewhat different goal. More recently it has been revisited in a similar sens as seen in this section in [5]. *Computer algebra systems* such as *Maple* [14] or *Mathematica* [21] can solve Equation 7 symbolically.

3.2 Notes for Using Equation 9

Let us consider the term κ given in Equation 8 within the context of Equation 9. Apparently, we must allow intermediate results to be in the realm of the complex domain $\mathbb{C}$ (results that represent physical properties such as v or t are strictly in the reals). In anticipation of forthcoming results compare to the values of κ in Table 3. For the reminder of this paper we will allow those intermediate results to be in the complex numbers (see any textbook on complex analysis such as [2]). Even though complex numbers and computation with those is supported for many platforms natively (e.g. for .NET since version 4) or by libraries (e.g. [9] for Java), it might not be a viable solution for all platforms. Wende [20] discusses the distinction of cases and a solution for remaining strictly in the reals.

The constant T is to be determined by *boundary conditions*, i.e. let an initial speed v_0 for $t = 0$ be given then T is computed by equation 9. From there the speed v for any time t where the domain is restricted by

$$\left|\Re\left(\frac{1}{2}\sqrt{-\kappa}(t+T)\right)\right| < \pi/2 \tag{10}$$

can be determined ($\Re(x)$ is the real part of x). This is not a restriction for our use case since $\lim_{x\to\pm\pi/2}\tan(x) = \pm\infty$ which implies that the range of $v(t)$ is equal to the whole set of reals $\mathbb{R}$.

3.3 Further Relations

The solution as given in Equation 9 doesn't suffice to perform the necessary computations. We will derive further relations in this section. If $\gamma = 0$ the differential Equation 7 and its solution becomes fairly simple and we do not go into further details. The concepts which we are going to discuss can be applied to those cases, as well. Note, that the equations for one of the simpler cases are equivalent to those used by Euler's approximation method.

Now, besides having the speed in dependency of $v(t)$ as in Equation 9 we need to determine t in dependency of v: $t(v)$, e.g. when we compute the time t_i up to the following speed limit v_i.

$$t + T = \frac{2}{\sqrt{-\kappa}} \arctan \frac{\beta + 2\gamma v}{\sqrt{-\kappa}} \tag{11}$$

Note that the inverse as in Equation 11 is not unique in general. The correct branch has to be chosen according to Equation 10. This is the case for many of the forthcoming equations and we will not mention the matter again since in practice it is rather straight forward to implement a correct solution.

Next, we gain formulas to bring the traversed distance s in relation to the time t. To this end Equation 9 is integrated which results in Equation 12

$$s = -\frac{\beta t + 2\ln\cos\psi}{2\gamma} \tag{12}$$

where

$$\psi = \frac{1}{2}\sqrt{-\kappa}(t + T). \tag{13}$$

Unfortunately, we did not succeed in gaining a closed representation of $t(s)$ from Equation 12 as it was possible in the previous paragraph for Equation 9. It might not even exist but is nevertheless required[1] to answer questions like "what is the time required until the end of the current segment is reached assuming constant parameters". To handle such computations we use a numeric root-finding algorithm. In the implementation we rely on *Brent's Method* [4] also known as the *Brent-Dekker Algorithm* [7] which combines various other methods and is heuristically known to be much faster then e.g. the *Bisection Method*. Note, that applying numerical root-search is not to be seen in an approximative sense as e.g. using Euler's Method to gain numerical solutions to a Differential Equation. Computers use a fixed precision[2] hence no loss in precision is imposed by root-finding in practice, see e.g. [1] for a general discussion.

We have collected the formulas to perform basic running time computations. From a regulatory point of view and to adhere to certain given rules also limits of acceleration $a = \dot{v}$ and the jerk $j = \ddot{v}$ are of concern [20]. The former is gained by applying the derivative to Equation 9 which results in Equation 14

$$a = \frac{-\kappa \sec^2\psi}{4\gamma}. \tag{14}$$

[1]precisely: answering $t(s)$ is required, the closed form is not

[2]which can be extended (arbitrarily for practical reasons) at the expense of memory and execution time

The latter can be derived by applying the derivative to Equation 14 and reads as in Equation 15

$$j = \frac{\sqrt{-\kappa}^3 \sec^2\psi \tan\psi}{4\gamma}. \tag{15}$$

Note, that Equation 14 can be reduced into explicit form $t(a)$ easily; however, for the jerk we recommend using root-finding. In both cases the inverse form is not unique with respect to sign and branch. Handling those correctly is not difficult but must be handled during implementation.

Finally, we mentioned that optimizing energy consumption is one of the main areas of application of this work. Note, that the mechanical work can be determined e.g. by evaluating

$$W = \int \vec{F}(t) \cdot \vec{v}(t) \, \mathrm{d}t \tag{16}$$

which can be carried out through elementary means using Equations 6, 9 and 14.

We collected all necessary relations to perform an algorithmic calculation of the running times of trains by an exact solution of the underlying problem. Apparently, the computations are slightly more involved compared to using approximative methods. The question is whether these computations are too expensive in efficiency (execution time). This is one of the central results of this work which we will answer in the forthcoming sections.

4 A Generic Algorithm for the Computation of the Running Time

4.1 Definition of the Problem

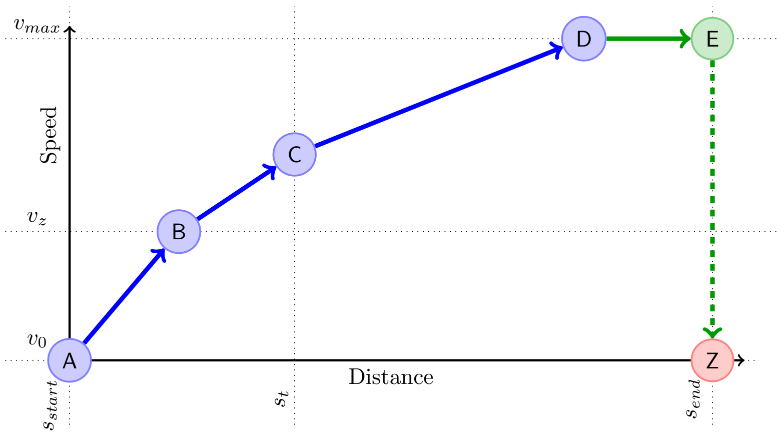
In time table optimization the goal is to minimize the time required to traverse a given track whilst adhering to certain constraints, such as speed limits. A basic algorithm for computation in a simplified model presents itself immediately: starting from some point with v=0 one proceeds forward in time until hitting an unsatisfied speed-limit from where deceleration computation is performed until the intersection is found. This procedure is repeated until the end point with $v = 0$ is reached. We will give a more precise description in the context of this work in the following paragraphs.

4.2 A Generic Algorithm

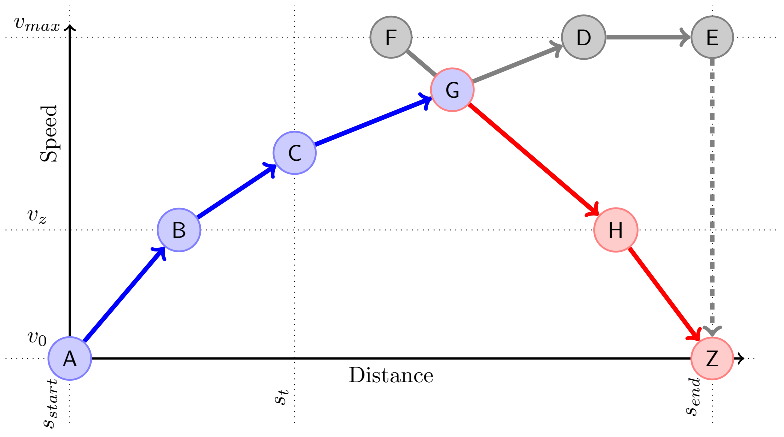
There are essentially three phases to be handled during a running time computation:

1. acceleration,
2. holding speed, and
3. deceleration.

We will now show how these work together with the foundations covered in Section 3. We employ an exemplary approach supported with figures rather than a formal pseudocode description. The illustrations in Figure 3 are symbolic, e.g. a segment is represented as a straight line instead of a function according to Equations 9 and 12. Compare to Figure 4 for a diagram based on a computation.



(a) Acceleration-Phase



(b) Deceleration-Phase

Figure 3: Algorithm

Acceleration

The acceleration-phase is illustrated in Figure 3(a). Assume we start out with an initial speed of $v = 0$, time $t = 0$ at the Point A. We pick the corresponding segment according to our current speed from the pull-force description, e.g. from Equation 4. Then the resistance parameters according to the composition-parameters (rolling-stock) and the track-parameters are added. This leaves us with a second order polynomial description for the current segment with limiting parameters v_{max}, s_{max} etcetera. We then compute T using Equation 11. Next, the limiting parameters for the current segment are considered and appropriately applied to compute $t_{\mathsf{A}\rightarrow\mathsf{B}}$. In the case from point A to point B this would be the speed limit v_z, e.g. given by the engine force. Now we have finished the first segment and the variables $t(s(\mathsf{B}))$ and $v(s(\mathsf{B}))$ are available as the start-condition for the following segment.

The computation of the segment $\mathsf{B} \rightarrow \mathsf{C}$ works similarly. Upon comparing the conditions it is determined that the segment is limited by the distance s_t, e.g. due to a change in parameters of the track. As mentioned in Section 3.3 we use numeric root-finding to compute the corresponding $t(\mathsf{B})$ from $s(\mathsf{B}) = s_t$. Given that, the speed $v(\mathsf{B})$ can be computed by Equation 9.

Finally, the last section $\mathsf{C} \rightarrow \mathsf{D}$ is just computed similarly to $\mathsf{A} \rightarrow \mathsf{B}$. However, at point D the general speed limit v_{max} is reached after which we switch to holding the speed.

Before we do so, let us mention other constraints. In case that there are limits on the acceleration a and or the jerk j the current segment must be further broken up according to Equation 14 and 15. The corresponding parts of the segment are then traversed according to the limiting constant a or j.

Holding the Speed

Two cases must be considered when the currently allowed maximum speed v_{max} is reached.

If the engine can provide more than the required force to keep up v_{max} the speed is kept until topology or other conditions require a reevaluation. This is the regular case, it is shown between the points $\mathsf{C} \rightarrow \mathsf{D}$ in Figure 3(a).

If not so, than the same principle as in the acceleration-phase will be used. However, a deceleration takes place, i.e. v at the end of the segment is lower than v_{max}.

Deceleration

Figure 3(b) add the deceleration-phase. Upon arrival at point E it is discovered, that the condition on v is not met. However, Once the point G is found, the procedure is again similar to an acceleration since the break force is modeled in the same way. There are two obvious ways to find the location of point G.

Starting from point Z we can “compute backwards” and check for each segment $\mathsf{Z} \leftarrow \mathsf{H}$, $\mathsf{H} \leftarrow \mathsf{F}$ if a cut-condition with the existing segments $\mathsf{A} \rightarrow \mathsf{B} \rightarrow \cdots \rightarrow \mathsf{E}$ is met. If so, the cut-point G is computed, the corresponding segments $\mathsf{C} \rightarrow \mathsf{D}$ and $\mathsf{H} \leftarrow \mathsf{F}$ are broken up, and connected as shown in Figure 3(b).

As a second method we can use regular forward computation in conjunction with numeric rood-finding along the segments $\mathsf{A} \rightarrow \mathsf{B} \rightarrow \cdots \rightarrow \mathsf{E}$ with the target condition that the speed at the target distance is met, e.g. $v = 0$ at $s(\mathsf{Z})$.

4.3 Notes

Constraints
Obviously, there are many variants and heuristics to speed up the algorithm or to accommodate further regulatory requirements. For example some countries mandate that the velocity must be kept constant for a certain amount of time between accelerating and decelerating. This can be easily done by using root-finding instead of backwards computation. However, discussing this in detail and the possibilities to include other various demands is out the scope of this work.

Energy Conservation
We declared the minimization of time as the primary goal for computing the running time as the basis for the outlined algorithm. Next, the usage of energy might be of concern. Assume e.g. that between two stop points a minimal time t_{min} has been computed. However, a time $t_x > t_{min}$ satisfies the condition given for operation. Then there are two ways to decrease energy consumption:

1. decreasing the maximum speed v_{max} according to Equation 2 (and in particular due to the completely lost $E = \frac{1}{2}\, m\, v^2$ if no recuperation is available), and

2. using coasting in conjunction with breaking and the usage energy recuperation in the deceleration phase, see [13, 22].

The first method can be simply applied by regarding the complete computation from $\mathsf{A} \rightarrow \mathsf{Z}$ as a function of v_{max} and use numerical root-finding with v_{max} as the variable and target condition $t_{total} = t_x$. Now matter if the second method is applied too, a fast computation enables optimization, either by a tractable algorithm or approximative e.g. by a gradient method with simulated annealing. We shall see that our proposed method is very fast compared to traditional methods in the following section.

5 Results

We implemented the outlined algorithm for running-time computation and conducted experiments with respect to execution time.

5.1 Notes to the Implementation, Runtime and Hardware

We implemented the algorithm depicted Section 4 in two variants:

1. The first variant is based on the solution of Equation 7 as described in Section 3.

2. The second variant is based on approximating the solution of Equation 7 with *Eulers Method*.

We have implemented both variants as similar as possible to provide a reliable relative comparison. If not otherwise stated the resolution used with Eulers Method was one second. This has been described as being sufficient [10]. We used *Scala* [16] as the programming language. It is a statically typed, object oriented and functional hybrid language for the

Table 1: Parameters of the Rolling-Stock

Property	**Value**
Mass [t]	507
Rotational Mass Equivalent [t]	24.5
Brake-Force [kN]	596.6
Resistance-Parameter r_0	7122
Resistance-Parameter r_1	0.0
Resistance-Parameter r_2	13.0

Table 2: Segments

Point	**Distance [m]**	**Time [s]**	**Speed [m/s]**
1	0	0	0
2	481	42.5	200/9
3	2209	97.0	350/9
4	8848	230.6	58.34
5	9515	244.3	350/9
6	9853	255.3	200/9
7	10000	268.5	0

Java Virtual Machine[3]. The compiled code was executed with SUN's 1.6 version of the Java-runtime on a machine with a 2.3 GHz Intel CPU running a 64bit Linux based operating system.

5.2 Rolling Stock Parameters of the Composition and Sample Track Definition

We reuse the engine force as given in Section 2.3. Table 1 shows the remaining physical parameters of our virtual composition. The parameters are derived from existing rolling stock data. However, in practice they are usually not composed in this way. We have chosen this composition to achieve exactly one interesting yet simple to verify example. When decelerating we assume that the (negative) engine force amends the force given by the brakes.

As a sample track we use a 10km long distance with no other parameters than $v = 0$ at the start and the end, compare to points A and Z in Figure 3. Also, we don't set any limits on the acceleration a (Equation 14) or jerk j (Equation 15). This gives a simple model with predictable results that can be easily verified.

Results of Traversal

Figure 4 shows a graphical representation how the track is traversed under the given conditions. Table 2 shows the numeric values of speed and Table 3 gives the numeric parameters

[3]an (at the time of writing outdated) CLR/.NET compiler for Scala exists too

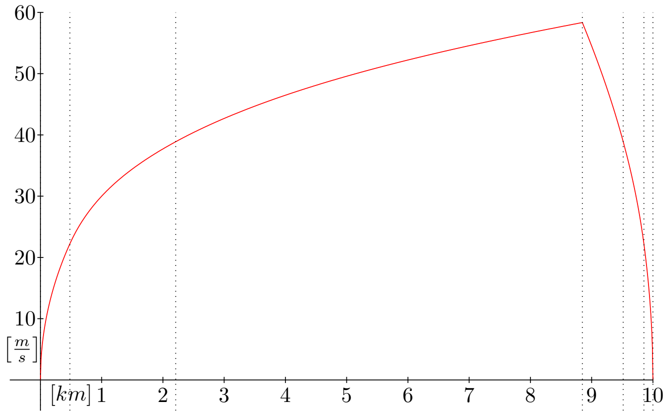


Figure 4: Speed-Distance Diagram for the Traversed Track

Table 3: Parameters of the Segments

Segment	κ	α	β	γ
1→2	$5.857 \cdot 10^{-5}$	$5.511 \cdot 10^{-1}$	$-2.117 \cdot 10^{-3}$	$-2.454 \cdot 10^{-5}$
2→3	$-4.220 \cdot 10^{-4}$	$1.353 \cdot 10^{0}$	$-5.128 \cdot 10^{-2}$	$5.639 \cdot 10^{-4}$
3→4	$-6.876 \cdot 10^{-5}$	$7.837 \cdot 10^{-1}$	$-2.108 \cdot 10^{-2}$	$1.637 \cdot 10^{-4}$
4→5	$-1.201 \cdot 10^{-3}$	$-1.933 \cdot 10^{0}$	$2.108 \cdot 10^{-2}$	$-2.127 \cdot 10^{-4}$
5→6	$-3.505 \cdot 10^{-3}$	$-2.502 \cdot 10^{0}$	$5.128 \cdot 10^{-2}$	$-6.130 \cdot 10^{-4}$
6→7	$-1.624 \cdot 10^{-4}$	$-1.700 \cdot 10^{0}$	$2.117 \cdot 10^{-3}$	$-2.454 \cdot 10^{-5}$

Table 4: Statistics of 100 Executions in Milliseconds

	DEQ Solution	Euler	Factor
Min.	2050	179550	
1. Qu.	2095	181002	
Median	2125	182240	85.76
Mean	2129	182534	85.74
3. Qu.	2153	184273	
Max.	2276	186597	

according to the formulas as given in Section 3.

5.3 Execution-Time

To average out influences from the system we composed the track as seen Figure 4 to 1500 copies and this computation was again repeated 100 times. The results with respect to the execution time can be seen in Table 4 and a graphical representation is given in Figure 5. Our implementation of the exact method performs approximately 85 times faster than the implementation of Eulers-Method.

Discussion

The resulting speed-up factor can depend to some extend on the implementation, programming language, runtime and even on the hardware. We already mentioned, that we tried to implement fairly as possible (note again, that the Euler Method is part of the exact solution when polynomial parameters take a simple form, see Section 3). We did not use any of the advanced functional paradigms, such as (tail-) recursion, in our implementation. We took advantage of some basic functional concepts which are available in object oriented programming too (less concise and more awkward though). In such cases compiled Scala code is known to be on par with compiled Java code [16]. Finally, our implementation is not optimized for fast execution in any way. As with the influence from the choice of the programming language, both methods, Euler and the exact, are affected. Since we compare relatively such effects should cancel out. Clearly, more restrictions on the track will introduce more segments which should affect the precise method more severely. However, a comprehensive evaluation is out of the scope of this work.

6 Conclusion and Future Prospects

We have shown that an implementation of the precise solution of the differential equation of the dynamics of trains outperforms traditional approximative methods, with respect of measurement of experimental execution time. We have argued were such an improved execution time can be valuable; namely in the optimization of train table schedules, and in particular in the conversation of energy. An implementation of the given method in conjunction with such optimization efforts would be the logical next step to take.

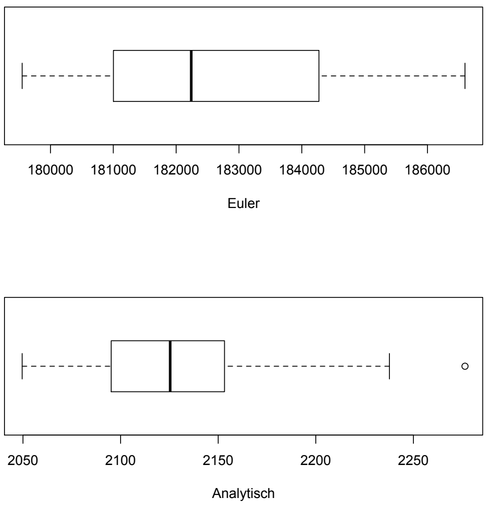


Figure 5: Box-Plots Corresponding to Table 4

7 Acknowledgments

We thank Dr. Julian D Yeandel for reviewing the manuscript. We further thank the reviewers for their constructive remarks.

References

[1] Kendall Atkinson. *An Introduction to Numerical Analysis*. Wiley, 2nd edition, 1989.

[2] Joseph Bak and Donald J. Newman. *Complex Analysis*. Springer, 3rd edition, 2010.

[3] William E. Boyce and Richard C. DiPrima. *Elementary Differential Equations and Boundary Value Problems*. Wiley, 7th edition, 2001.

[4] R. P. Brent. Algorithms for minimization without derivatives. 1973.

[5] Olaf Brünger and Elias Dahlhaus. Running time estimation. *Timetable & Traffic*, pages 58–82, 2008.

[6] W.J. Davis. The tractive resistance of electric locomotives and cars. *General Electric Rewiew*, 29, 10 1926.

[7] T. J. Dekker. Finding a zero by means of successive linear interpolation. In In B. Dejon and P. Henrici, editors, *Constructive Aspects of the Fundamental Theorem of Algebra*. Wiley-Interscience, 1969.

[8] Gerhard Dobner and Hans-Jürgen Dobner. *Gewöhnliche Differenzialgleichungen*. Fachbuchverlag Leipzig, 2004.

[9] Michael Thomas Flanagan. Michael thomas flanagan's java scientific library - complex class: Complex arithmetic and complex functions. http://www.ee.ucl.ac.uk/ mflanaga/java/Complex.html, 2010.

[10] Daniel Hürlimann. *Objektorientierte Modellierung von Infrastrukturelementen und Betriebsvorgängen im Eisenbahnwesen*. Schriftenreihe IVT Nr. 125, ETH Zürich, 2002. Dissertation.

[11] Annegret Liebers, Dorothea Wagner, and Karsten Weihe. On the hardness of recognizing bundles in time table graphs. In Peter Widmayer, Gabriele Neyer, and Stephan Eidenbenz, editors, *WG*, volume 1665 of *Lecture Notes in Computer Science*, pages 325–337. Springer, 1999.

[12] E. Lindelöf. Sur l'application de la méthode des approximations successives aux équations différentielles ordinaires du premier ordre. In *Comptes rendus hebdomadaires des séances de l'Académie des sciences.*, volume 114, pages 454–457. 1894.

[13] Piotr Lukaszewicz. *Energy Consumption and Running Time for Trains*. PhD thesis, KTH Royal Institute of Technology Stockholm, 2001.

[14] maplesoft. Maple. http://www.maplesoft.com/products/Maple/.

[15] Matthias Müller-Hannemann, Frank Schulz, Dorothea Wagner, and Christos Zaroliagis. Timetable information: Models and algorithms. In Frank Geraets, Leo Kroon, Anita Schoebel, Dorothea Wagner, and Christos Zaroliagis, editors, *Algorithmic Methods for Railway Optimization*, volume 4359 of *Lecture Notes in Computer Science*, pages 67–90. Springer Berlin / Heidelberg, 2007.

[16] Martin Odersky, Lex Spoon, and Bill Venners. *Programming in Scala*. Artima, Mountain View, CA, 2008.

[17] Leon W.P. Peeters. *Cyclic railway timetable optimization*. PhD thesis, Erasmus Universiteit Rotterdam, 2003.

[18] G. Strahl. Verfahren zur bestimmung der belastungsgrenzen von dampflokomotiven. *Zeitschrift des Vereines deutscher Ingenieure*, 57, 02 1913.

[19] Jörg von Lingen (coordinator). railml 2.0. http://railml.org, 2009.

[20] Dietrich Wende. *Fahrdynamik des Schienenverkehrs*. Teubner, 2003.

[21] Wolfram-Research. Mathematica. http://www.wolfram.com/products/mathematica/.

[22] S. Yasukawa, S. Fujita, T. Hasebe, and K. Sato. Development of an on-board energy-saving train operation system for the shinkansen electric railcars. In *QR of RTRI*, volume 28, 1987.

[23] Daniel Zwillinger. *Handbook of Differential Equations*. Academic Press, 3rd edition, 1997.