Jan Přikryl; Miroslav Vaniš

Comparing numerical integration schemes for a car-following model with real-world data

In: Jan Chleboun and Pavel Kůs and Petr Přikryl and Karel Segeth and Jakub Šístek and Tomáš Vejchodský (eds.): Programs and Algorithms of Numerical Mathematics, Proceedings of Seminar. Janov nad Nisou, June 19-24, 2016. Institute of Mathematics CAS, Prague, 2017. pp. 89–96.

Persistent URL: http://dml.cz/dmlcz/703002

Programs and Algorithms of Numerical Mathematics 18
J. Chleboun, P. Kůs, P. Přikryl, K. Segeth, J. Šístek, T. Vejchodský (Eds.)
Institute of Mathematics CAS, Prague 2017

COMPARING NUMERICAL INTEGRATION SCHEMES FOR A CAR-FOLLOWING MODEL WITH REAL-WORLD DATA

Jan Přikryl, Miroslav Vaniš

Czech Technical University in Prague, Faculty of Transportation Sciences
Konviktská 16, CZ-110 00 Praha, Czech Republic
prikryl@fd.cvut.cz, vanismir@fd.cvut.cz

Abstract: A key element of microscopic traffic flow simulation is the so-called car-following model, describing the way in which a typical driver interacts with other vehicles on the road. This model is typically continuous and traffic micro-simulator updates its vehicle positions by a numerical integration scheme. While increasing the order of the scheme should lead to more accurate results, most micro-simulators employ the simplest Euler rule. In our contribution, inspired by [1], we will provide some additional details that have to be addressed when implementing higher-order numerical integration schemes for CFMs and we will show that the theoretical gain of higher-order methods is unfortunately masked out by the stochastic nature of real-world traffic flow.

Keywords: numerical integration, Runge-Kutta, Euler, trapezoid, ballistic update, car-following model, intelligent driver model, traffic flow

MSC: 65L06, 65L07, 68Q17

1. Introduction

In time-continuous car-following models (CFMs) employed by some microscopic traffic flow simulators, the acceleration of individual vehicles is described by a function of the driver's characteristic behavior and the surrounding traffic. This formulation leads to a coupled set of ODEs which is identical to that of physical particles following Newtonian dynamics with the physical forces replaced by fictious "social forces". While many CFMs have been formulated directly in discrete time in the form of difference equations, or fully discretely as cellular automata, and can be therefore evaluated directly, time-continuous CFMs must be evaluated using a numerical integration scheme in all but the most trivial analytically solvable cases [3].

The interaction between different vehicles on the road is best described by a process where vehicles react primarily on the driving behaviour of their leaders, i.e. vehicles that drive in front of the modelled vehicles [5]. If formulated in continuous

DOI: 10.21136/panm.2016.11

form, this class of CFMs is often represented by a coupled system of ODEs modelling movement of a single vehicle,

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = f(\mathbf{y}, t), \tag{1}$$

where $\mathbf{y} = [x, v]^{\mathrm{T}}$ is the state vector of a vehicle, composed of position x and speed v, and the vector function f represents the specific CFM. A wide class of follow-the-leader CFMs that are used in microscopic simulators can be defined using a vehicle acceleration function a^{mic} as

$$\begin{aligned} \frac{\mathrm{d}x_i}{\mathrm{d}t} &= v_i, \\ \frac{\mathrm{d}v_i}{\mathrm{d}t} &= a^{\mathrm{mic}}(s_i, v_i, v_{i-1}), \end{aligned} \tag{2}$$

where $i = 1, \ldots, n$ is the index of actual vehicle from the fixed n number of vehicles, x_i denotes the position of the front bumper of vehicle i, v_i its speed and s_i bumper to bumper gap between the current vehicle and its leader (for the first vehicle we have $s_1 = \infty$, otherwise $s_i = x_{i-1} - x_i - \ell_{i-1}$, where ℓ_i is the length of the i-th vehicle). By convention, we assume that for the i-th vehicle, the vehicle $i - 1$ is the leader.

1.1. Intelligent driver model

In their recent article [1], Treiber and Kanagaraj compare different numerical integration schemes for *intelligent driver model* (IDM) [2]. The IDM is a time-continuous car-following model for the simulation of freeway and urban traffic, developed in part by the first author of [1]. The model assumes that drivers maintain certain minimal time and space gap from their leading vehicle, and at the same time try to keep their preferred speed and acceleration and deceleration profile. The model does not work with an explicitly given reaction time of the driver, and typically it is assumed that the driver reaction time is constant and equal to the step size of the underlying numerical integration scheme.

The IDM acceleration function is given by

$$a^{\mathrm{mic}}(s_i, v_i, v_{i-1}) = a \left(1 - \left(\frac{v_i}{v_0} \right)^{\delta} - \left(\frac{s^*(v_i, v_{i-1})}{s_i} \right)^2 \right) \tag{3}$$

with

$$s^*(v_i, v_{i-1}) = s_0 + v_i\, T + \frac{v_i(v_i - v_{i-1})}{2\sqrt{ab}}, \tag{4}$$

where the model parameters are a, b – maximum safe acceleration and deceleration, v_0 – preferred speed, δ – acceleration exponent (typically equal to 4), and s_0, T – minimum space and time gap to the leader vehicle.

2. Numerical methods

Similarly to the authors of [1] we will investigate four different integration schemes. The first three are well-known numerical methods for integrating ODEs, namely

- explicit Euler method, defined as

$$\begin{aligned} \mathbf{k}_1 &= f(\mathbf{y}, t), \\ \mathbf{y}(t+h) &= \mathbf{y} + h\mathbf{k}_1, \end{aligned} \tag{5}$$

- explicit trapezoidal rule (Heun's method)

$$\begin{aligned} \mathbf{k}_1 &= f(\mathbf{y}, t), \qquad \mathbf{k}_2 = f(\mathbf{y} + h\mathbf{k}_1, t + h), \\ \mathbf{y}(t+h) &= \mathbf{y} + \frac{h}{2}(\mathbf{k}_1 + \mathbf{k}_2), \end{aligned} \tag{6}$$

- and the standard fourth-order Runge-Kutta method (RK4)

$$\begin{aligned} \mathbf{k}_1 &= f(\mathbf{y}, t), \qquad \mathbf{k}_2 = f\left(\mathbf{y} + \frac{h}{2}\mathbf{k}_1, t + \frac{h}{2}\right), \\ \mathbf{k}_3 &= f\left(\mathbf{y} + \frac{h}{2}\mathbf{k}_2, t + \frac{h}{2}\right), \qquad \mathbf{k}_4 = f(\mathbf{y} + h\mathbf{k}_3, t + h), \\ \mathbf{y}(t+h) &= \mathbf{y} + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k_4}). \end{aligned} \tag{7}$$

Besides the three standard approaches outlined above, the authors of [1] use an alternative first-order integration scheme called *ballistic update*, which can be used only in special cases where Eq. (1) represents Newtonian dynamic acceleration equations. The rule can be interpreted as a mixed first-order, second-order update consisting of an Euler update for the speeds, and a trapezoidal update for the positions,

$$\mathbf{y}(t+h) = \begin{pmatrix} \mathbf{x}(t+h) \\ \mathbf{v}(t+h) \end{pmatrix} = \begin{pmatrix} \mathbf{x} \\ \mathbf{v} \end{pmatrix} + h \begin{pmatrix} \mathbf{v} \\ a(\mathbf{x}, \mathbf{v}) \end{pmatrix} + \frac{1}{2}h^2 \begin{pmatrix} a(\mathbf{x}, \mathbf{v}) \\ \mathbf{0} \end{pmatrix}, \tag{8}$$

where the last term makes the difference between ballistic and Euler method. The acceleration is computed only once per the time step, so the order of the method stays the same as in the Euler method. The trapezoidal rule needs to calculate the acceleration two times and the fourth order Runge-Kutta four times. For CFMs, calculating the acceleration function is an essential part of their numerical complexity.

3. Comparing the numerical integration methods

In order to compare the performance of integration schemes introduced in Section 2 we need to have a reference solution of the studied problem. For all the schemes, both decreasing the time step and increasing the method order should lead to more accurate results at the price of higher computational demands. We will therefore evaluate the difference to the reference solution as a function of the computational effort that was spent at obtaining the tested solution.

3.1. Reference solution

The reference solution of a simulation involving IDM cannot be obtained analytically, except for trivial cases without vehicle interaction. We will follow the argumentation of the original paper and generate a reference solution for simulation scenarios using RK4 and time step $h_{\text{ref}} = 10^{-4}\,\text{s}$. The global discretisation error will be evaluated for speeds of a single vehicle as an average of local absolute errors,

$$\epsilon = \left\| v_i^{\text{num}} - v_i^{\text{ref}} \right\| = \frac{1}{m} \sum_{j=1}^{m} |v_i^{\text{num}}(jh) - v_i^{\text{ref}}(jh)|, \tag{9}$$

where $v_i^{\text{num}}(jh)$ is the speed of the i-th vehicle at time $t = jh$ and $v_i^{\text{ref}}(jh)$ is the reference solution for the same vehicle at the same time step.

3.2. Numerical complexity

The computational demands of a given integration scheme can be expressed in terms of *numerical complexity* as

$$C = \frac{p}{h}, \tag{10}$$

where p denotes the number of evaluations of the acceleration function for one step of the integration scheme (for Euler method and ballistic update $p = 1$, for trapezoidal rule $p = 2$, for RK4 we have $p = 4$) and h is the time step.

4. Implementation details

Reference [1] contains almost complete information needed to re-implement the original experiments of the authors. We will now briefly overview the important parts, adding one detail that has been omitted from the original paper.

4.1. Initial and boundary conditions

For the original (synthetic) simulation scenarios the following conditions hold: At time 0 s all vehicles are stopped, $v_i(0) = 0$, and their positions are $x_1(0) = 0$, $\forall i > 1 : x_i(0) = x_{i-1}(0) - \ell_{i-1} - s_i$. Furthermore, a boundary condition for the first vehicle acceleration is given – Treiber and Kanagaraj assume free-flow conditions for the first vehicle,

$$a_1(v_1, t) = a_{\text{free}}(v_1) = a^{\text{mic}}(\infty, v_1, v_1). \tag{11}$$

These conditions lead to an autonomous ODE.

For the real world scenario we have vehicles that are entering the simulation at externally prescribed time instants τ_i, and hence we have $v_i(\tau_i) = v_{0,i}$ and $x_i(\tau_i) = 0$. Special care has to be taken to keep the gap between the entered vehicle and its leader large enough.

5. Heuristics for stopping vehicles

The discussed integration schemes assume smooth f. Due to finite update times, all equations will lead to negative speeds in cases where the vehicle stops between the updates. The authors of [1] suggested the following heuristics to estimate the stopping position directly: when the computed speed after the final step of integration scheme is negative, the position of the stopped vehicle is determined by a variant of the ballistic rule instead of the originally calculated position:

$$x_i(t + \widetilde{h}) = x_i(t) - \frac{v_i^2(t)}{2a_i^{\mathrm{mic}}(t)}. \tag{12}$$

Here, $\widetilde{h}$ could be h or $h/2$ (for the RK4 method). In addition, the speed of the vehicle is reset to zero.

5.1. Position of the leader

Except for the Euler method and the ballistic update, the integration of IDM for the current vehicle requires the knowledge of the leader vehicle state $\mathbf{y}_{\mathrm{lead}}(t+h)$ (for the computation of $\mathbf{k}_2$ of trapezoidal rule and $\mathbf{k}_4$ of RK4) and, for RK4, also the leader vehicle state at the intermediate point $\mathbf{y}_{\mathrm{lead}}(t+h/2)$, which has to be used to correctly compute values of $\mathbf{k}_2$ and $\mathbf{k}_3$. While the former value can be easily obtained by performing the integration step on an ordered sequence of vehicles, thus updating the state of the leader vehicle before updating the state of its follower, the original paper does not mention how the state $\mathbf{y}_{\mathrm{lead}}(t + h/2)$ is computed.

We have tested simple linear approximation, trapezoidal rule, and different combinations of intermediate RK4 states. Our results, presented in Section 2, indicate that in [1] the state at the intermediate point is computed as

$$\mathbf{y}_{\mathrm{lead}}\left(t + \frac{h}{2}\right) = \mathbf{y}_{\mathrm{lead}}(t) + \frac{h}{2}\left(\mathbf{k}_2 + \mathbf{k}_3\right). \tag{13}$$

5.2. Parallelisation

Note that while the Euler and the ballistic updates may be executed on arbitrary vehicle regardless of the updated state of its leader, the ordering condition for trapezoidal and RK4 update effectively prevents vehicle-level parallelisation. However, vehicles within independent lanes may be still updated in parallel.

6. Results

To verify the results presented in [1] and to compare these results with real-world measured data, a simple custom micro-simulator has been created in Python programming language, using standard extension libraries NumPy, SciPy and Matplotlib. The simulator is able to simulate a single lane road equipped with vehicle counting detectors and provide floating car data (speed, position, acceleration, and gap-to--leader) for all simulated vehicles.

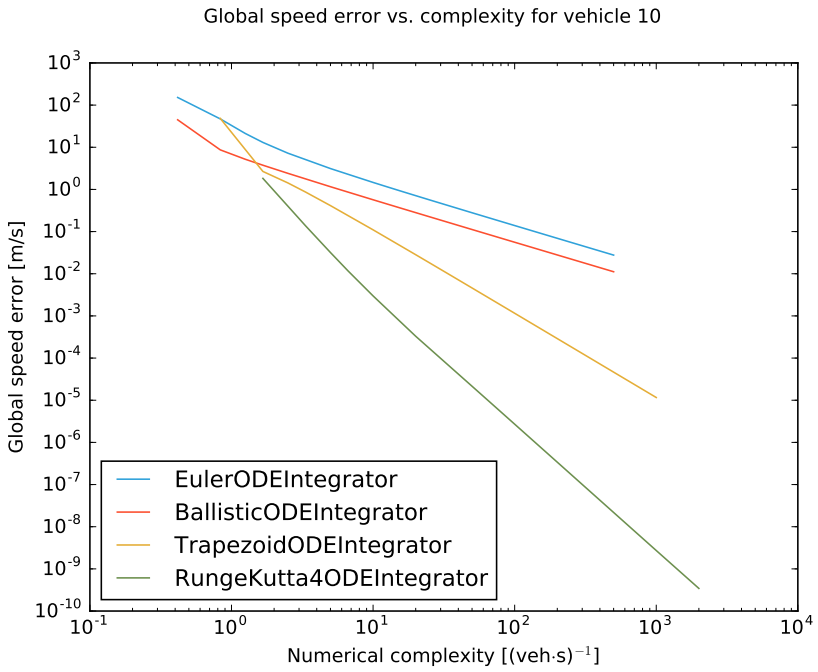

Figure 1: Global discretisation error of the 10-th vehicle speed as a function of the numerical complexity for the four update schemes. As with the original paper [1], the simulation interval has been limited to $[0, 60]$ seconds.

6.1. Original scenario

In order to verify the implemented integration schemes, we first replicated the original synthetic start-stop scenario of Treiber and Kanagaraj. We started with 20 identical vehicles queuing at the red light. At time $t = 0$, traffic light turned green and the queued cars started moving for 670 metres, where the next signalized intersection was located and the cars needed to stop again. We ran the simulation of this scenario for all four numerical integration schemes mentioned in Section 2 for 16 different integration steps ranging from 2.4 s to 0.002 s. If implemented correctly, we would expect to observe results similar to that of [1], with RK4 being the most precise of the tested methods in terms of the L_1 global error metric (9). This is indeed true in case that the intermediate point in RK4 scheme is computed using Eq. (13), see Fig. 1.

6.2. Real world data

In order to test the results on real world data, the same data set from the southern leg of the Prague Ring (SOKP) as in Ref. [4] has been used. From the database of weekday measurements, two working days have been selected. One day of detector measurements for passenger vehicles from SOKP gantry at km 20.1 has been used to calibrate IDM parameters. Then, another working day has been used as an input

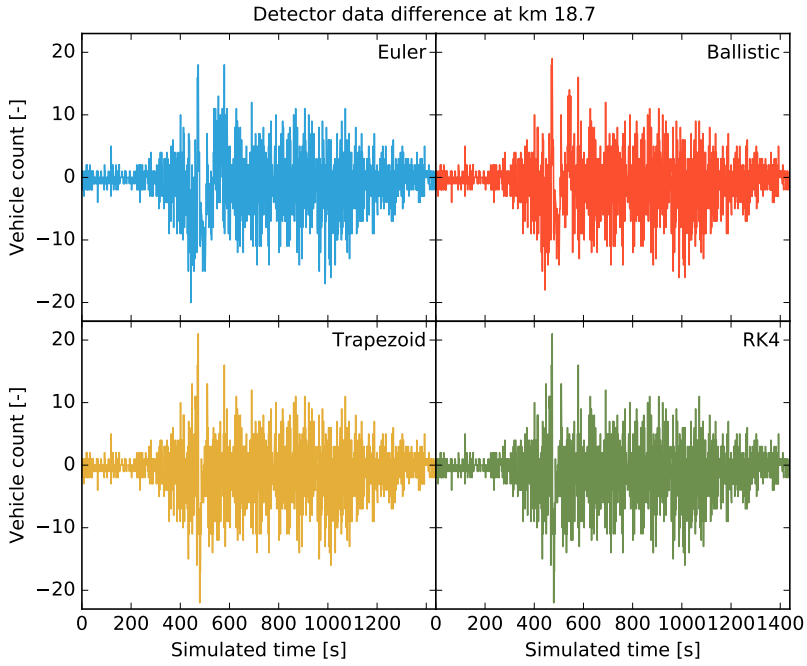

Figure 2: Global error of the vehicle count for the four update schemes with respect to real measurements.

to vehicle generator and the simulation result has been compared to data measured at km 18.7. The fixed step $h = 0.5\,\mathrm{s}$ (that roughly corresponds to average driver reaction time) has been used for all four integration schemes. As we can see in Fig. 2, results for the different integration schemes are almost identical. When we compute different error metrics, the similarity of the results becomes even more obvious – see Tab. 1.

We believe these results confirm the fact the even a well-accepted model as the IDM is only an approximation of the reality (which is to greater or lesser extent true for all mathematical models). The traffic flow is inherently a phenomenon with a very strong stochastic component, and as such it is very difficult to model – even if we would be able to replicate the properties of every vehicle with a minimal error, it is impossible to predict the behaviour of a human driving the vehicle.

7. Conclusions

Generally, when integrating ODEs, the fourth-order Runge–Kutta (RK4) method is the de-facto standard and other methods are rarely used. However, this is not the case for integrating CFMs in traffic simulations – here, Euler's method is still the most widespread one [1]. One of the reasons is given by the authors of [1]: for typical traffic-related situations, RK4 cannot reach its theoretical consistency order $p = 4$ as the smoothness conditions for the integrated function are rarely satisfied.

Method	MSE	$\max \lvert\Delta\rvert$	# hits
Euler	20.88	20	259
Ballistic	19.73	19	258
Trapezoid	20.19	22	262
RK4	20.67	21	261

Table 1: Comparison on real world data. MSE stands for mean squared error, $\max \lvert\Delta\rvert$ is the maximum difference from the reference value, and # hits column contains the count of occurrences where the simulation results were within ± 1 vehicle from the reference.

Inspired by this observation, which was made using synthetic experiments, we have used the same group of integration schemes to simulate real traffic between two measurement points on a highway. Our result show that due to the stochastic nature of traffic, the performance of all integration schemes is almost identical, suggesting that using Euler's method (or ballistic update) can be justified by its low computational demands.

The reader could correctly object that selecting only a single step size, namely $h = 0.5\,\mathrm{s}$, for comparison, may be unfair to higher order methods as their benefits would become more pronounced for larger h. Unfortunately, in CFM context, h also often reflects the average reaction time of a driver and its choice is therefore limited to values between circa 0.5 and 1 second.

The Python source code of our experiments is available from the GitHub of the fist author at http://github.com/jprk/panm18.

References

[1] Treiber, M. and Kanagaraj V.: Comparing numerical integration schemes for time-continuous car-following models. Phys. A **419** (2015), 183–195.

[2] Treiber, M., Hennecke, A., and Helbing, D.: Congested traffic states in empirical observations and microscopic simulations. Phys. Rev. E **62** (2000), 1805-–1824.

[3] Treiber, M. and Kesting, A.: *Traffic flow dynamics: data, models and simulation.* Springer, Berlin, 2013.

[4] Horňák, I. and Přikryl, J.: Experimental comparison of traffic flow models on traffic data. Programs and Algorithms of Numerical Mathematics **17** (2015), 86–91.

[5] Gazis, D. C., Herman, R., and Rothery, R. W.: Nonlinear follow-the-leader models of traffic flow. Oper. Res. **9** (1961), 545–567.