Technical Assessment : Traffic Modeling with Car-Following

George Gunter - Lab of Daniel Work

February 4, 2020

1 Introduction

The goal of this document is to walk you through a series of technical assessments revolving around the simulation of car-following models. This assessment goes along with a jupyter notebook that can be downloaded here. It is highly recommended (probably even necessary) that each portion of the notebook be completed in order.

2 Topic 1: Simulating a single vehicle

This portion assessment focuses on the basics of simulating single vehicles in a traffic stream. We'll work with the simple case of one vehicle following another vehicle, whose path is predetermined.

2.1 Introduction to car-following

Car following models have been a rich area of research for over 50 years, with initial experimental work being done by General Motors [1]. Since then they have become a standard method for the simulation of traffic streams in every area of traffic, from freeway driving to the urban network.

In general a car-following model consists of an *ordinary differential equation* (ODE) that is a function of the target (to be simulated) vehicles speed (v), inter-vehicle spacing (s), and the difference in speed between it and a vehicle in front of it (Δv). Using these quantities the vehicle's acceleration is modeled. Generally this can be written via:

$$f(v, s, \Delta v) = \frac{dv}{dt} \tag{1}$$

Using this dynamical model we then wish to then find the corresponding speed and spacing profiles which the target vehicle will take over time. This is done via solving the ODE.

In general there are many methods for "solving" differential equations in order to recover lower derivative states from the higher derivative function, or in other words how to get v and s from $f(v, s, \Delta v)$. In this example we're going to just use a naive Euler forward stepping method that can be written as such:

$$\begin{aligned}
&a(t_k) = f(v(t_k), s(t_k), \Delta v(t_k), \\
&v(t_{k+1}) = v(t_k) + \Delta T(a(t_k), \\
&s(t_{k+1}) = s(t_k) + \Delta T \Delta v(t_k) = s(t_k) + \Delta T(v_l(t_k) - v(t_k))
\end{aligned} \tag{2}$$

where t_k is the time at the discrete sample number k, a is acceleration, v_l is the leading vehicle's speed, and ΔT is the time-step distance. This is a numerical solution of the ODE, rather than an analytic one. This method is ill-conditioned (not accurate) for large ΔT, but is often sufficient for ΔT under .5 seconds or so. In this example we'll use a ΔT of .1 seconds. It's interesting to note that ΔT can to some extent be thought of as a kind of reaction time for the driver, in that if the driver is more attentive than ΔT will be smaller, while larger ΔT may result in less steady driving.

2.2 Simulation using the Bando Optimal Velocity model

Let's now look at a specific car-following model, the Bando Optimal Velocity (OV) model. This model is originally proposed in [2] due to its ability to capture interesting traffic flow patterns. The version that we'll work with can be written as such:

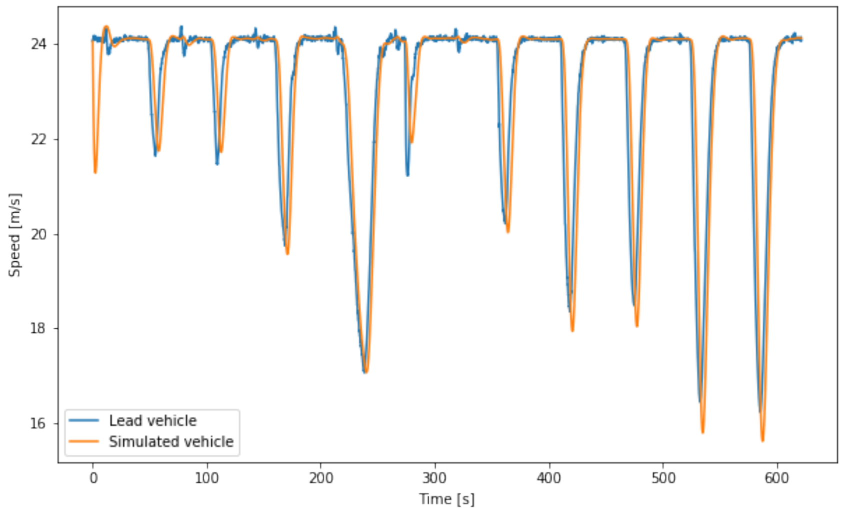
$$\begin{aligned} f(v,s),\Delta v &= \alpha(V(s)-v)+\beta(\tfrac{\Delta v}{s^2}), \\ V(s) &= V_m * (\tfrac{tanh(\frac{s}{s_0})-tanh(s^*)}{1+tanh(s^*)}) \end{aligned} \tag{3}$$

where α and β are gain parameters for the two portions of the equation, one of which is the optimal velocity function ($V(s)$) and the other is a relative velocity term. V_m,s_0, and s^* all determine properties of the optimal velocity function.

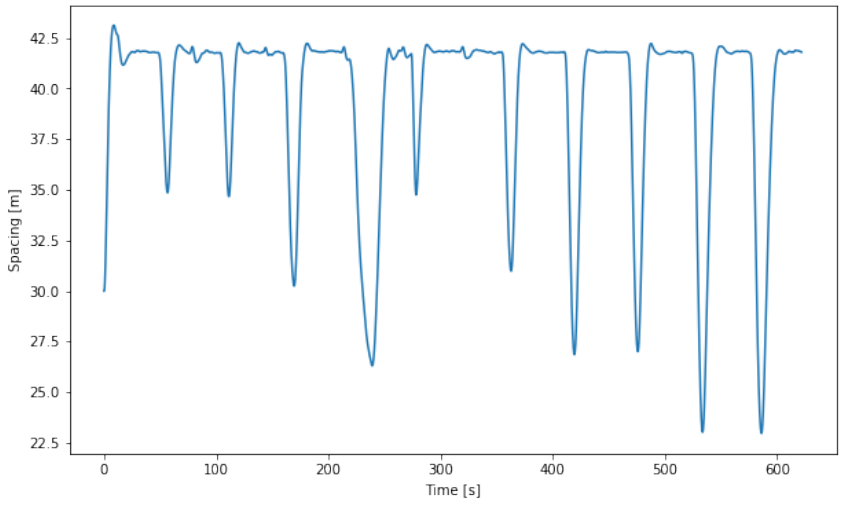
Let's use this model and equation (2) to simulate a driver responding to a lead vehicle which executes a series of breaking and speeding up maneuvers. The lead driver data contains time sample values in one column and the vehicle's speed in another. The fidelity of the data is 10 herz, meaning ΔT is .1 seconds. Using parameter values of:

$$\begin{aligned} &\alpha = .5, \\ &\beta = 20.0, \\ &s_0 = 10.0, \\ &s^* = .5, \\ &V_m = 30.0 \end{aligned} \tag{4}$$

and initial conditions of $v(0) = v_l(0)$ and $s(0) = 30$ we get a speed profile as such:



and a spacing profile as such:

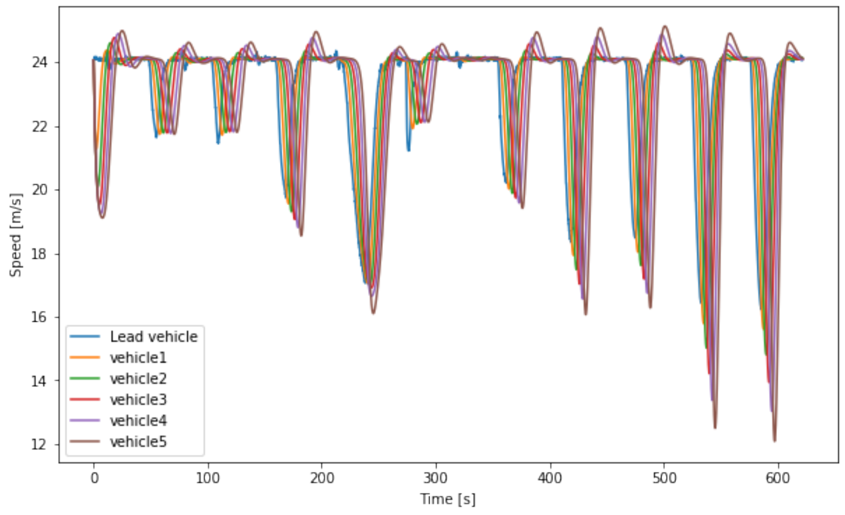


Try to make sure your results are similar to the above results before moving on to the next section. Do you see any problems with our simulation? What might be worth changing?

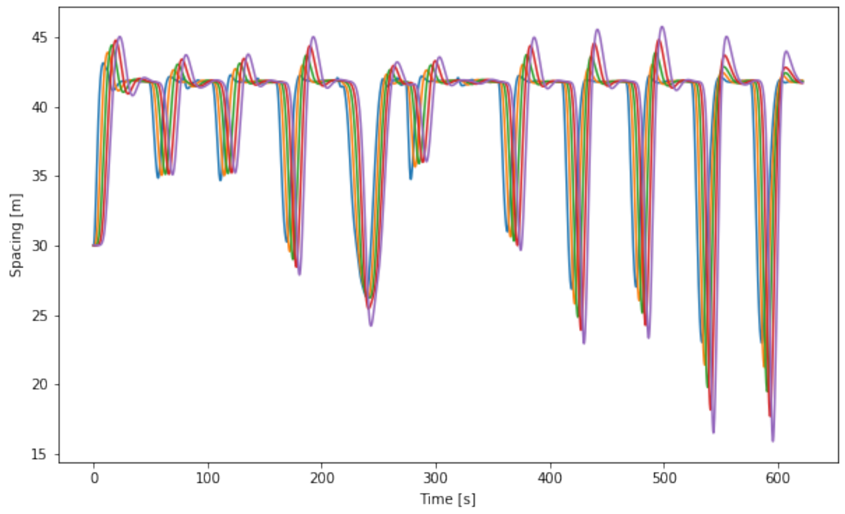
3 topic 2: Simulating a string of vehicles

In this section we're going to take the simulator that we developed in section 2 and use it to simulate a string of vehicles, all following the original lead vehicle's trajectory. Fundamentally, nothing really changes. Each vehicle will still simply determine its acceleration from a leading vehicle's speed and the spacing to it. However, this time each vehicle, except for the very first simulated vehicle and the lead, will need to look to another simulated vehicle for state information.

Let's try first simulating a string of 5 vehicles. We're going to use the same initial conditions and parameters that we used for the first case, but for all vehicles now. This is known as a *homogeneous* platoon. Plotting their speeds we get:



and for the spacing we get:



What do you notice about the emergent properties of the vehicles as a traffic stream? What appears to happen if we were to simulate many vehicles following one another? Try playing around with the number of vehicles simulated. Another way of visualizing the data that may be of interest is via space-time diagrams. Here's a website to help you visualize how they're made.

Make sure that you've completed through at least the end of section 2 before moving on.

4 Topic 3: Model Calibration

In this section we will explore how recorded data from an driver can be used to find good models for reproducing that driving behavior. This is generally known as *car-following model calibration*, which is a vital part of micro-simulation. More broadly, it can be thought of as an instance of system-identification, in which we seek to learn the underlying dynamics of a system when presented with time-series measurements. In this example we'll use one possible approach to solve the problem, but many exist.

4.1 Problem formulation

In order to find good models for drivers we may want to use collected data by measuring the behavior of drivers over a time. In general this data is hard to come by, but some examples exist, such as the NGSIM dataset and the HighD [3] dataset, which tracker driver's positions over time. This data can then be used to extract speed, spacing, and relative speed measurements, which provide the elements necessary to evaluate a car-following model, such as that in equation (3). Another dataset, the one that will be used in this example, was collected by our lab in 2018 and measures the car-following behavior of several brands of *Adaptive Cruise Control* (ACC) vehicles [4].

In order to use this data optimally we now form an optimization problem which we will solve using repeated simulation. Such an approach is (intuitively) called *simulation-based optimization*. The problem can be formulated as such:

$$\begin{aligned}
\text{minimize :}\quad & e(\boldsymbol{\theta}, v^{\mathrm{m}}, s^{\mathrm{m}}, v_l^{\mathrm{m}}) \\
\text{subject to:}\quad & \dot{v}(t) = f(\boldsymbol{\theta}, s, v, \Delta v) \\
& \dot{s}(t) = v_l(t) - v(t), \\
& v(0) = v^{\mathrm{m}}(0) \\
& s(0) = s^{\mathrm{m}}(0) \\
& \Delta v(0) = v_l^{\mathrm{m}}(0) - v^{\mathrm{m}}(0) \\
& \boldsymbol{\theta} \geq 0
\end{aligned} \tag{5}$$

where $e(\theta, v^{\mathrm{m}}(0), s^{\mathrm{m}}, v_l^{\mathrm{m}})$ is an objective function that calculates some error for a given parameter set, $\boldsymbol{\theta}$, and the measured data, $v^{\mathrm{m}}, s^{\mathrm{m}}$, and v_l^{m}. The constraints then detail both what the dynamics for the simulation are, and how initial conditions should be specified. Additionally, while potentially not strictly speaking necessary for all car-following models, the $\boldsymbol{\theta} \geq 0$ tells us that for any given potential set of parameters none of them should be below 0. For most car-following models, negative parameters lead to non-physical behavior (i.e. driving through other cars, or accelerating to infinity). See here for more information about unstable dynamical models.

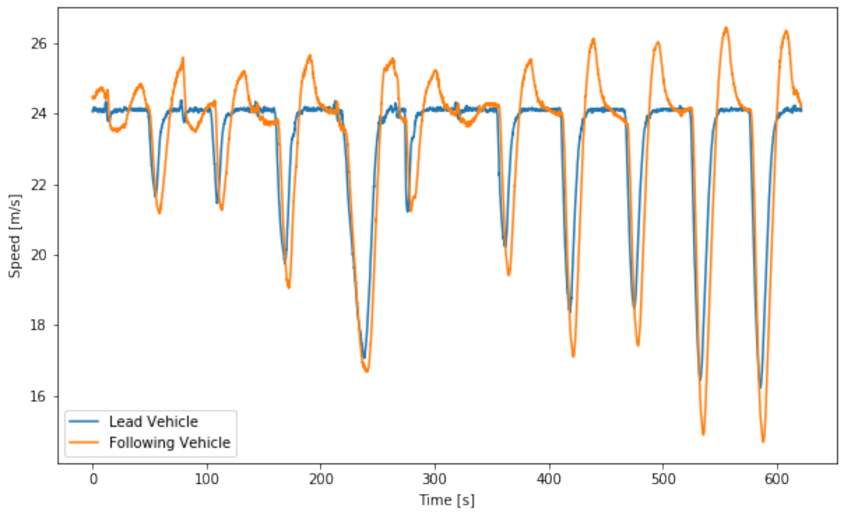
To concretely solve this problem for our case let's consider the following error function:

$$e(\boldsymbol{\theta}, v^{\mathrm{m}}, s^{\mathrm{m}}, v_l^{\mathrm{m}}) = \sqrt{\frac{1}{T}\int_0^T (v^{\mathrm{m}}(t) - v(t))^2 dt} \tag{6}$$

This is a *root-mean-squared error* (RMSE) between a simulated speed profile ($v(t)$), and the recorded ($v^{\mathrm{m}}(t)$). The task now is to find a $\boldsymbol{\theta}$, such that an in simulation our model optimally reproduces the recorded data.

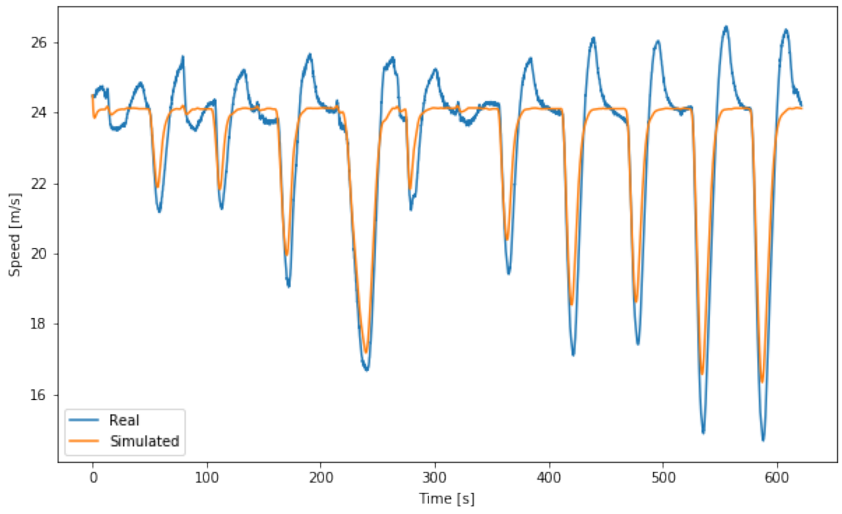
4.2 Solving the problem

First, let's look at what the data looks like:



In general we can note that the following vehicle is trying to drive the same speed as the lead vehicle, but overshoots on accelerations, and undershoots on braking. Does that seem desirable?

Now we need to actually solve the problem. In this case since the optimization problem is constrained we need to use a constrained method. Read here for more about constrained optimization. In this example, I solve the problem by using the python package `scipy.optimize`, which has a general optimization solver, `minimize`. In particular I used the trust-constraint method from a starting point of $\boldsymbol{\theta} = [\beta = 10, \alpha = .1, v_m = 40, s_0 = 20, s^* = .2]^T$. The end result I get is that of $\boldsymbol{\theta} = [1.5, .034, 32, 38, .047]^T$. When comparing this with the recorded data we get:

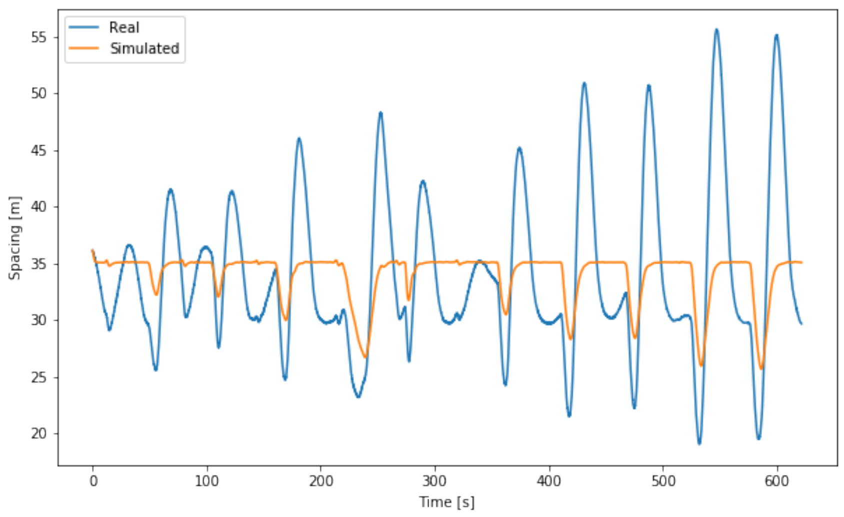


Does this seem like a sound fit? In what ways to it seem to capture the right behavior and in what ways not? What might we do to try and do better?

Try changing your error metric to one which is spacing based:

$$e(\boldsymbol{\theta}, v^{\mathrm{m}}, s^{\mathrm{m}}, v_l^{\mathrm{m}}) = \sqrt{\frac{1}{T}\int_0^T (s^{\mathrm{m}}(t) - s(t))^2 dt} \tag{7}$$

Try the same calibration routine but now with this different objective function. I get a new model of the form $\boldsymbol{\theta} = [2.5, 9.5, 52, 37, 0.87]$. This gives a spacing in simulation that compares to the recorded spacing as such:



How do these two models compare to one another? Is it better to use speed or spacing as our error metric? Should we be using both at once? Are either of these methods good enough?

References

[1] D. C. Gazis, R. Herman, and R. B. Potts. Car-following theory of steady-state traffic flow. *Operations Research*, 7(4):499–505, 1959.

[2] M. Bando, Hesebem K., A. Nakayama, A. Shibata, and Y. Sugiyama. Dynamical model of traffic congestion and numerical simulation. *Physical Review E*, 51(2):1035–1042, 1995.

[3] Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2118–2125, 2018.

[4] George Günter, Derek Gloudemans, Raphael E. Stern, Sean T. McQuade, Rahul Bhadani, Matt Bunting, Maria Laura Delle Monache, Roman Lysecky, Benjamin Seibold, Jonathan Sprinkle, Benedetto Piccoli, and Daniel B. Work. Are commercially implemented adaptive cruise control systems string stable? *CoRR*, abs/1905.02108, 2019.