Traffic Models using Continua and Cellular Automata

Author: Peter Johanson May 2, 2004

0.1 Question

This paper analyzes the differences found in analyzing traffic models with continua using a discretization of PDEs and using a cellular automata approach for individual vehicles. In particular, a linear and non-linear update model will be compared. Ultimately, having a better knowledge of the small scale car models, larger models of rotaries, complete city systems, etc can more "accurately" mimic real life (this accuracy is a largely subjective thing). The focus of this paper was to be on the influence of right-of-way rules in rotaries, but time constraints and a need to have a more "mathy" paper has caused the topic to be constrained to the specifics of the two speed functions analyzed herein.

0.2 Conservation of Cars

Analysis of traffic usually springs from one immutable equation:

$$\rho_t + (\rho v)_x = 0 \tag{1}$$

Where ρ is the density of cars, and $v(\rho)$ is a car speed dependent on density. This may alternatively be written as

$$\rho_t + F_x = 0 \tag{2}$$

where $F = \rho v(\rho)$ is the flux. The choice of $v(\rho)$ is ultimately the interesting factor in the discussion. The two models to be discussed will be a linear function of ρ and one using $\tanh$.

0.3 Discretization vs. Cellular Automata

The paradigm shift between discretization and cellular automata is accomplished through the realization that $\rho \propto \Delta x$ where Δx is the distance between two cars. Below are the two functions we will use in our analysis:

$$u_1(\rho) = u_0 \left(1 - \frac{\rho}{\rho_0}\right) \tag{3}$$

$$u_2(\rho) = u_0 \tanh\left(\frac{\rho_0}{\rho} - 1\right) \tag{4}$$

where u_0 is the maximum car velocity, and ρ_0 is the maximum density of cars. These use of tanh is relatively arbitrary, but both models share an important fact:

$$u_1(\rho_0) = u_2(\rho_0) = 0 \tag{5}$$

$$u_1(0) = u_2(0) = u_0 \tag{6}$$

This fact is important, as it ensures that the velocity models mimic "real life" in as much as a car will stop moving when it's in heavy traffic, and accelerate to it's maximum speed given “the open road.” Figure 1 shows the two models as functions of ρ. These two functions can be rewritten for the cellular automata version as so:

$$u_1(\Delta x) = u_0 \left(1 - \frac{\Delta x_0}{\Delta x}\right) \tag{7}$$

$$u_2(\Delta x) = u_0 \tanh\left(\frac{\Delta x}{\Delta x_0} - 1\right) \tag{8}$$

where u_0 is the maximum car velocity and Δx_0 is the minimum distance between two cars.

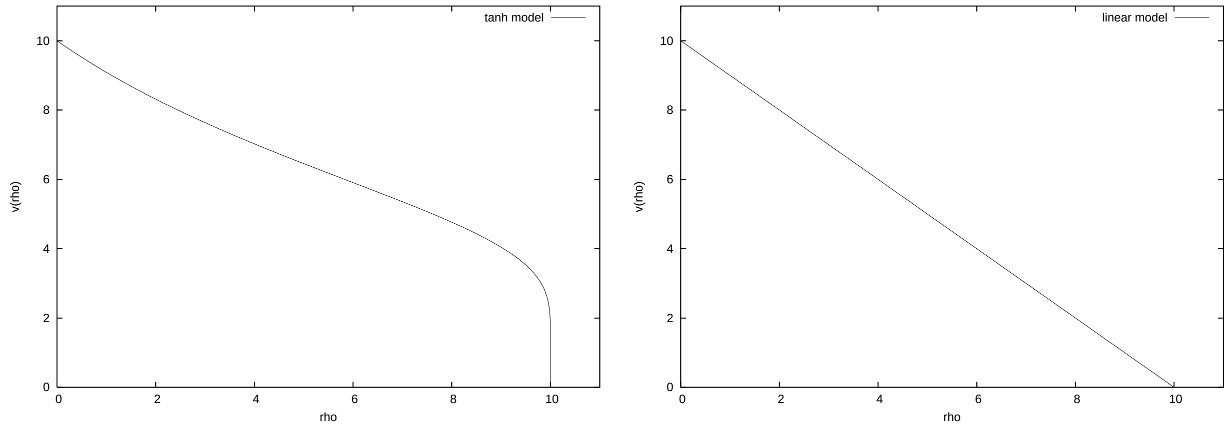


Figure 1: Two car speed models

0.4 Cellular Automata Implementation

Implementing a cellular automata version is considerably easier than a continua model. In particular, the issue of discretization in the x direction are no longer a concern, and we need only worry about accuracy based on the discretization in time. However, along with this freedom, we are also burdened with needing to worry about car collisions. Each car has two indices indicating the car index, and the time slice. The location and speed of the j-th car at time index i is given by x_j^i and $(x')_j^i$ respectively. The process for finding x_j^{i+1} and $(x')_j^{i+1}$ is as follows:

- Set $x_j^{i+1} = x_j^i + (x')_j^i * \Delta t$.
- If $x_j^{i+1} \geq x_{j+1}^i + (x')_{j+1}^i * \Delta t$, handle a collision (described below).
- For $\Delta x = x_{j+1}^i - x_j^i$, if $\Delta x \leq \Delta x_0$, then $(x')_j^{i+1} = 0$, otherwise $(x')_j^{i+1} = u(\Delta x)$.

For $\Delta t \to 0$, car collisions will not be a problem, as our criteria for our model is that $u(\Delta x_0) = 0$ so cars will stop before hitting a vehicle in front of them. In reality, when approaching a stopped vehicle, if the time slice is not sufficiently large, the position will be updated with a large enough speed from the previous iteration that a car will "zip past" the car in front of it. To prevent collisions, a check is made to determine if the position of x_j^{i+1} is past the predicted x_{j+i}^{i+1}. If the new position will be a problem, we follow the following algorithm:

if $(x_j^{i+1} >= x_{j+1}^{i+1})$

if $(x_{j+1}^{i+1} - \Delta x_0 <= x_{j-1}^{i+1})$

$$x_j^{i+1} = \left(x_{j-1}^{i+1} + x_{j+1}^{i+1}\right)/2$$

else

$$x_j^{i+1} = x_{j+1}^{i+1} - \Delta x_0$$

end

end

If there's room, place the car at Δx_0 behind the car ahead, otherwise put it at the average between x_{j-1}^{i+1} and x_{j+1}^{i+1}.

0.5 Conversion to Burger's Equation

To analyze the linear model from a PDE standpoint, one converts the conservation law for cars into burger's equation. The transformation hinges on the recognition of part of our conservation being the flux:

$$\rho_t + (u\rho)_x = 0 \tag{9}$$

where $u(\rho) = u_0\left(1 - \frac{\rho}{\rho_0}\right)$. To convert, we'll start by finding F', and then defining a new wave speed c:

$$F' = u + \rho\frac{du}{d\rho} \tag{10}$$

$$= u_0\left(1 - \frac{\rho}{\rho_0}\right) - \frac{\rho}{\rho_0}u_0 \qquad = u_0\left(1 - 2\frac{\rho}{\rho_0}\right) \tag{11}$$

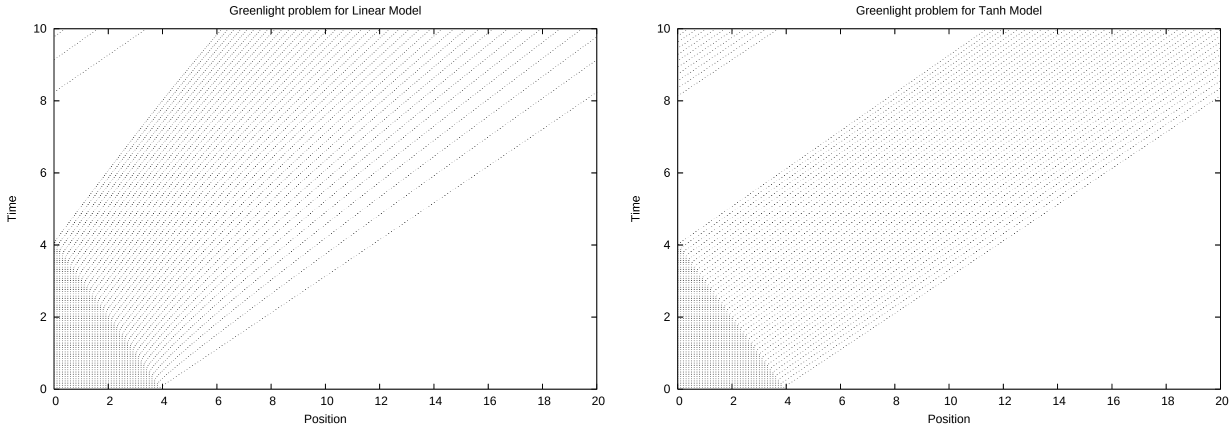


Figure 2: Sample run of greenlight problem

With this new knowledge of F', we can plug back into our conservation law for cars:

$$\rho_t + u_0\left(1 - 2\frac{\rho}{\rho_0}\right)\rho_x = 0 \tag{12}$$

We now define c equal to the coefficient of ρ_x in the above equation, and then we have a new definition for ρ:

$$\rho = -\left(\frac{c}{u_0} - 1\right)\frac{\rho_0}{2} \tag{13}$$

$$= \text{constant} - \frac{\rho_0}{2}c \tag{14}$$

This allows us to convert our original conservation law:

$$\rho_t + u_0\left(1 - 2\frac{\rho}{\rho_0}\right)\rho_x = 0 \tag{15}$$

$$\Rightarrow \frac{-\rho_0}{2u_0}\left\{c_t + cc_x\right\} = 0 \tag{16}$$

$$\Rightarrow c_t + cc_x = 0 \tag{17}$$

which is the familiar Burger's equation.

0.6 Discretization of Burger's

To discretize Burger's equation we can use finite volume methods to do approximations in the x direction. We'll use up-winding to ensure that we have better accuracy for our discretization. The code was based off of ddwave3.m found at http://cs.nyu.edu/ barnett/waves/ddwave3.m.

The idea is to use up-winding, along with finite volume methods to generate c_j^{n+1} based on the speed at the given location. As most of our analysis is on a rotary going on direction, most of the time this will always boil down to the same case.

The idea is this:

$$u_t + uu_x = u_t + F_x = 0 \tag{18}$$

In the x field, we consider small cells centered at index j with width Δx. We then can define

$$u_j = \frac{1}{\Delta x}\int_{j\text{th cell}} u dx \tag{19}$$

That means we can define $F = \frac{u^2}{2}$. Our initial version of Burger's can then be written as the following for the jth cell:

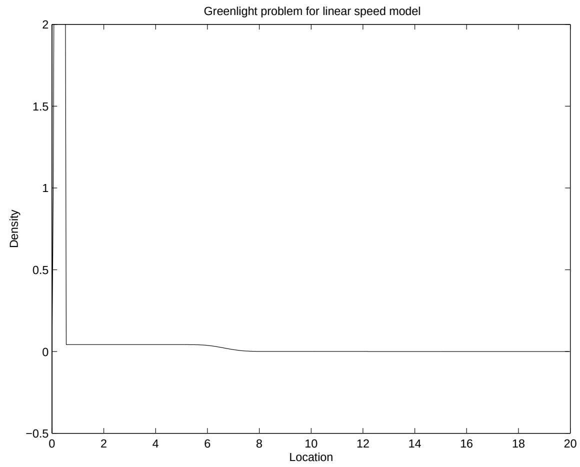


Figure 3: Greenlight Problem with linear speed model

$$\Delta x \frac{d}{dt} u_j + \int_{j\text{th cell}} F_x dx = 0 \tag{20}$$

$$\Rightarrow \Delta x \frac{d}{dt} u_j + \left(F_{j+\frac{1}{2}} - F_{j-\frac{1}{2}} \right) = 0 \tag{21}$$

We can then write a relation between u_j^i and u_j^{i+1} as such:

$$\frac{u_j^{i+1} - u_j^i}{\Delta t} + \frac{F_{j+1/2} - F_{j-1/2}}{\Delta x} = 0 \tag{22}$$

$$\Rightarrow u_j^{i+1} = u_j^i - \frac{\Delta t}{\Delta x} \left(F_{j+1/2} - F_{j-1/2} \right) \tag{23}$$

The interesting part them becomes determining what to use for the flux at these points. This is where the first order up-winding comes into play.

$\hat{c} = \frac{u_j^i + u_{j+1}^i}{2}$
if $\hat{c} > 0$

$$F_j = \frac{\left(u_j^i\right)^2}{2}$$

else

$$F_j = \frac{\left(u_{j+1}^i\right)^2}{2}$$

end
if $u_j^i < 0$ and $u_{j+1}^i > 0$

$$F_j = -\frac{u_{j+1}^i + u_j^i}{2}$$

end

We can then use F_j and F_{j+1} for our flux for u_j^{i+1}.

To test the greenlight problem, special care must be taken. In particular, the initial conditions for the greenlight problem are such that from 0 to x_0 you have $u_n = -u_{max}$ and from x_0 to N we have $u_n = u_{max}$. The end result is that the total flux is 0, so no change will occur. To solve this issue, we must add a small perturbation of our initial conditions. Care should be taken that this perturbation be small enough not to skew the initial conditions so much that the data is corrupted too much. A small change of 0.1 was used for testing.

Here is the result of using the linear speed function with the greenlight problem. The cars are packed at the maximum density and then set free. You can see that section of the large stacked cars is moving forward all at the same speed, causing a second lower plateau of cars to spreed to the right from the larger high plateau. This corresponds to the chunk of cars all advancing at the same speed at equal distances seen in Figure 2.

0.7 Continuum Model of Tanh Function

Similar methods can be taken to convert arbitrary functions $u(\rho)$ to Burger's equation. In particular:

$$rho_t + F_x = 0 \tag{24}$$
$$= rho_t + F' rho_x \tag{25}$$

Now we define $c = F'$ and multiply by c':

$$c' rho_t + c' c rho_x = 0 \tag{26}$$
$$= c_t + c c_x \tag{27}$$

In the particular instance where $F = (\rho u(\rho))$, $c' = F''$ is a nasty beast indeed. Beyond that, there is a complication caused by shocks in the system. The finite volume methods described previously rely heavily on the fact that shock behavior is determined greatly by conservation of cars. In the case where we have linearity, conservation of the integral of ρ is the same as conservation of the integral of c, so we can use the Burger's stuff fine, when we use tanh however, this does not hold true. As such, comparison with the PDE model will be restricted to the linear model.

0.8 Error

After the models have been implemented, it's important to ensure that we will have accurate results from the two models, and see which model requires smaller Δt to achieve good results. These error plots were generated by taking in the position of each car after a 10 second run with a $\Delta t = \frac{1}{200}$ with the position of the same car after a 10 second run with a $\Delta t = \frac{1}{400}$ We can see that the tanh model has significantly different locations for a higher time step in the error test of the greenlight model. This can attributed to the steep rise in the velocity function $u(\Delta x)$ for the tanh model as the distance between the cars increases.

0.9 Conclusion

Both models meet the mathematical criteria placed forth for a valid car model. Which model is "better" at modeling real life is a subjective and unquantifiable determination based on the topic of this study. None the less, the choice between these two models should be evaluated based on needs; the linear model is more easily converted to a continuous model, whereas the tanh model may be found to be more "realistic." Even more realistic models may be used if we take one step further and use functions of acceleration (see **Future Developments**) at the price of extra computation. Again, the extra power needed to do this need be weighed against the "realism" potentially gained.

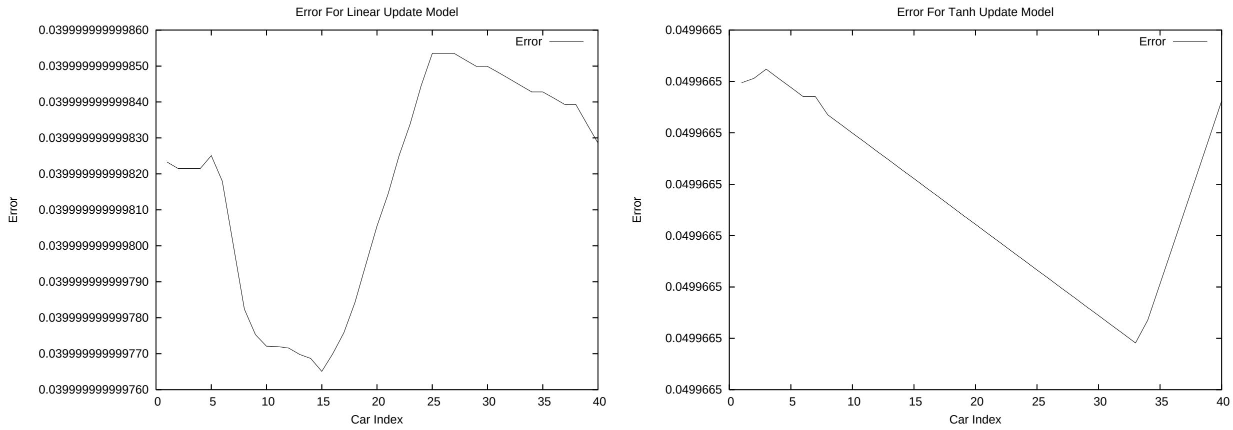
0.10 Future Developments

Beyond studying functions of speed based on some criteria, one might also study a cellular automata approach where the lookup function is used to determine the **acceleration** at each time step. This might result in even more accurate models, with the ability to accurately predict collisions, etc by defining x''_{min} and x''_{max} as the maximum and minimum accelerations respectively (we'll keep x''_{min} positive, and consider it the maximum deceleration). Combined with a x'_{max}, we would better be able to model different types of cars. (Imagine a speedy sports car with a large x''_{max} and large x'_{max} versus a Mack truck with a comparatively low x''_{max} and x''_{min} and a mid-range x'_{max}). Our update model then might be something like:

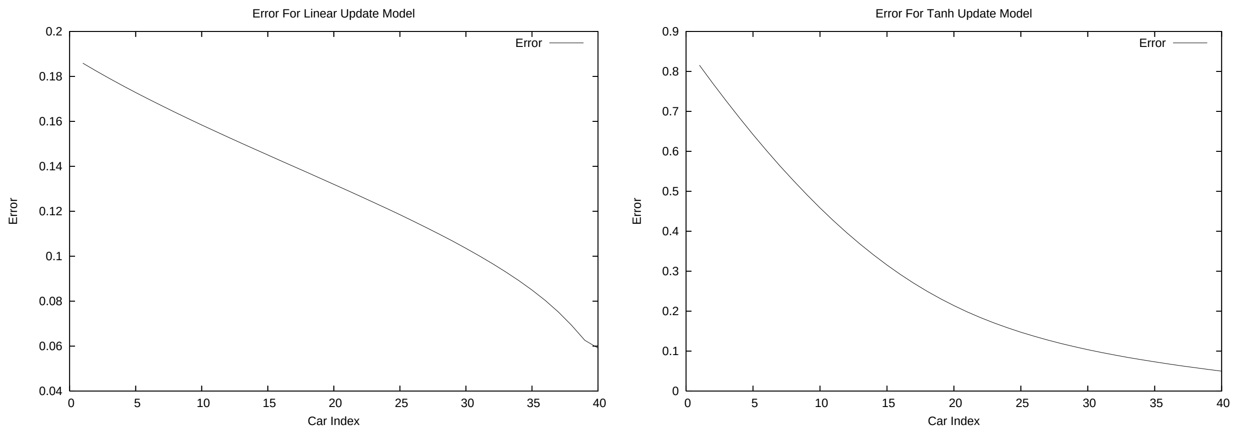
- Set $x_j^{i+1} = x_j^i + (x')_j^i * \Delta t$
- Set $(x')_j^{i+1} = (x')_j^i + (x'')_j^i * \Delta t$
- Set $(x'')_j^{i+1} = F\left(x_{j-1}^i, x_j^i, x_{j+1}^i\right)$

where $F\left(x_{j-1}^i, x_j^i, x_{j+1}^i\right)$ is a new function for determining acceleration based on the attributes of the car itself, and the cars behind and ahead of it from the previous time step.

The real goal of my studies is to analyze traffic rotaries, in particular how the right-of-way rules affect traffic throughput, the average time it takes to traverse a rotary, etc. Intuitively, one would think that having cars in a rotary



(a) Error for even distribution initial condition



(b) Error for greenlight distribution initial condition have right-of-way precedence over cars entering the rotary would result in fewer traffic jams, and a corresponding increase in throughput and decrease in average time through a rotary, but this may not in fact be the case.

Time constraints and a need to focus on immediate results for this paper has limited my opportunity to study these effect, but future work will hopefully continue to study these facts. My current work on creating a more general framework for these studies using an object-oriented programming language (which solves many of the limitations found in using Matlab for studies involving more than the function used for car speed) is available at http://pages.nyu.edu/ pjj205/TrafficModel/. This implementation is done in C#/GTK# and requires an installation of Mono and the gtk-sharp libraries found at http://www.go-mono.com.