Traffic Simulation Toolbox

User's Manual

Jesse Haber-Kucharsky
Shreyas Sundaram

University of Waterloo
Department of Electrical and Computer Engineering

May 31, 2011

Contents

1 Introduction

The Traffic Simulation Toolbox is a set of interfaces, classes, and functions for MATLAB which help to simulate the motion of a sequence of moving bodies (cars). Each individual car's motion is governed by a feedback control system in which the controller and the plant can be chosen arbitrarily. The Traffic Simulation Toolbox can be used to explore the effect of different models for the plant and the controller on the emergent behaviour of the sequence of cars.

The Traffic Simulation Toolbox consists primarily of a main GUI combining configuration, the simulation window, and some useful plots. This GUI should be sufficient for the vast majority of cases. However, should extra flexibility or programmability be required, a set of functions are provided which can be used independently of the GUI.

The first section of this manual, Section 2, discusses how to run basic simulations using the Traffic Simulation Toolbox main window and the included models.

The next section, Section 3, discusses how to add new models to the system, how to modify existing models, and how integrate them with the main GUI window.

2 Basic Use

This section discusses the basic use of the Traffic Simulation Toolbox. The following quick-start guide demonstrates the core functionality of the main GUI window.

2.1 Quick-Start Example

To run the main GUI window for the Traffic Simulation Toolbox, execute `trafficSimulation.m`. You should be presented with the window shown in Figure 1.

We are going to run a simulation involving ten cars, for five seconds, using the default settings for the individual cars and the default models for the control system. We will then change a parameter of the model dictating the behaviour of the cars and observe the result on the sequence of cars.

1. Adjust the `Simulation Time` field in the `Simulation` section of the main window to `5`. This adjusts the span of the simulation to five seconds.

2. Using the drop-down menu in the `Settings` section of the main window, adjust the `Number of Cars` to `10`. For now, we will accept the default car properties so ignore the button titled `Modify Car Properties`.

3. To begin the simulation, press the `Play` button in the `Simulation` section of the main window. At any time the simulation can be paused by pressing the `Pause` button and resumed by pressing the `Play` button again.

4. Once the simulation has completed it can be reset by pressing the `Reset` button in the `Settings` section of the main window. The simulation can then be run again, or the parameters can be changed.

After running the simulation (but before it is reset), the main window should look as it does in Figure 2.

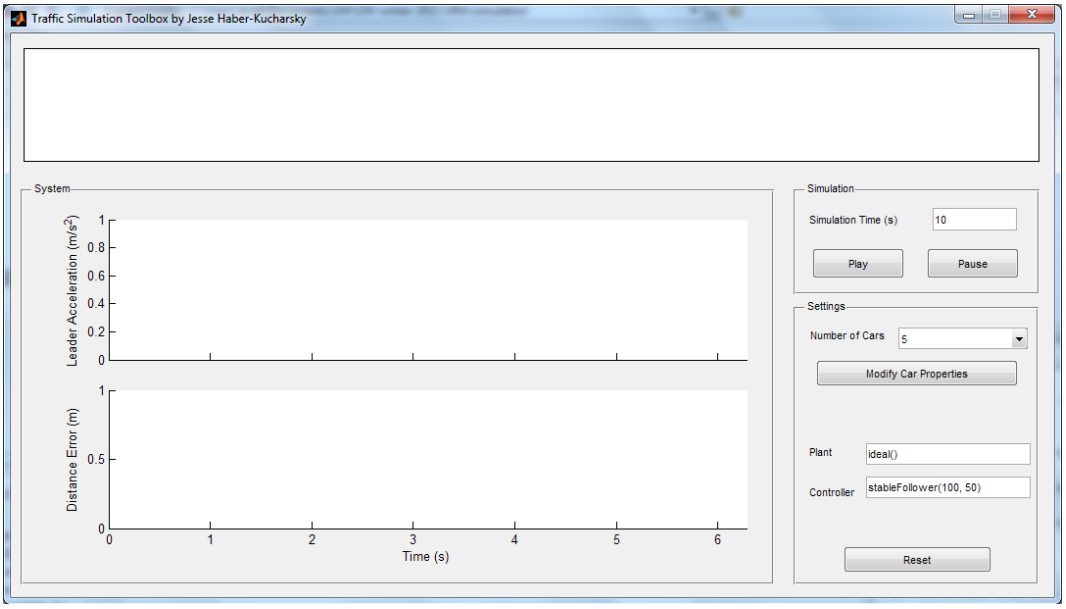

Figure 1: The main window of the Traffic Simulation Toolbox when it has been started.

Each of the black rectangles represents a car, and the red rectangle indicates the leader car. The leader car is the only car in the sequence which has its motion predetermined. More specifically, the leader car's acceleration can be any function of time. By default, the leader's acceleration is given by

$$a_{\text{lead}}(t_i) = 100 \sin(5t_i)$$

where t_i is the i$^{\text{th}}$ instant in time.

We can see the plot of this function in the upper plot of the main window.

The motion of each of the cars in the sequence is determined from the motion of the cars which precede it. The exact nature of this behaviour depends on the model chosen for the *controller* and *plant*. For the purposes of this quick-start guide, we can consider these to be mathematical models for a given car's motion (acceleration, velocity, and position) as a function of some other car's motion (the *leader*) and the current state of the car.

Note that every car in the sequence can be set to follow an arbitrary leader (provided that the leader car is ahead in the sequence). By default, each car's leader reference is the car immediately preceding it in the sequence.

In our simulation (and by default), the `stableFollower` model is used

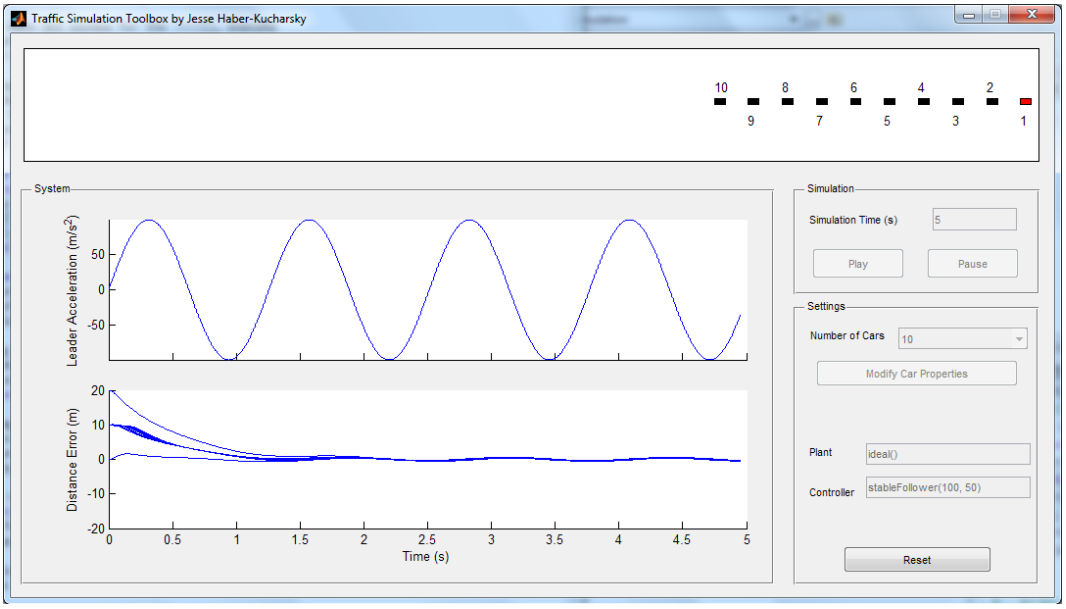

Figure 2: The main window of the Traffic Simulation Toolbox after the first simulation (described above) has completed.

as the controller. The `stableFollower` model calculates a car's acceleration at a given instant in time as follows:

$$a(t_i) = K_1 \left[p_L(t_i) - p(t_i) - D \right] + K_2 \left[v_L(t_i) - v(t_i) \right]$$

Conceptually, the acceleration of the car is a function of two errors: the error between the car's position and its ideal position and the error between the cars velocity and its ideal velocity.

$a(t_i)$ represents the acceleration of the car at a given time, t_i.

$p(t_i)$ and $p_L(t_i)$ represent the position of the car and the car's leader reference at t_i, respectively.

$v(t_i)$ and $v_L(t_i)$ represent the velocity of the car and the car's leader reference at t_i, respectively.

The constants K_1 and K_2 represent the gain associated with each error. For example, if K_1 is larger than K_2 then error in the separation of the cars will contribute more to the car's acceleration than error in the velocity of the cars.

Lastly, D represents the ideal separation between the cars. For cars which follow a leader immediately in front of them, this ideal separation is $D =$

$L_0 = 10\,\text{m}$. For cars with an arbitrary leader reference n cars away in the sequence, this separation is $D = nL_0$.

In our simulation, parameters for the `stableFollower` model are $K_1 = 100$ and $K_2 = 50$. This choice of parameters resulted in a stable system: the cars appeared to travel at uniform velocity and they maintained the ideal separation distances. We can see this information plotted in the lower axis. It shows the distance error (i.e. the difference between the ideal position of a car and its actual position) for a selection of cars in the sequence.

As time progressed in the simulation, we can observe that the error converged to approximately zero for all cars (though there was a slight sinusoidal pattern around zero).

Now, let's change the value of K_2 and observe the effect on the sequence of cars. By reducing the value of K_2, we might expect that the velocity error contributes less to the acceleration of the car.

1. If the simulation has not yet been reset, make sure to do so by pressing the `Reset` button in the `Settings` section of the main window.

2. Adjust the second parameter of `stableFollower` in the `Controller` field in the `Settings` section of the main window from `50` to `20`.

3. Press the `Play` button in the `Simulation` section of the main window to begin the simulation.

Once the simulation has completed, the main window should appear as it does in figure Figure 3.

We can observe that the distance error is more pronounced. More noticeable though is the wave that appears to propagate through the sequence of cars whenever the leader abruptly accelerates.

Clearly, these constants are not as desirable for maintaining a steady stream of traffic.

2.2 Modelling Cars as Feedback-Control Systems

Fundamentally, the Traffic Simulation Toolbox models the motion of each car as a feedback-control system as shown in Figure 4.

The motion of each car is represented as its time-domain acceleration, stored as a vector of discrete indexed values. The simulation time is stored similarly. The car's velocity and position are (approximately) determined

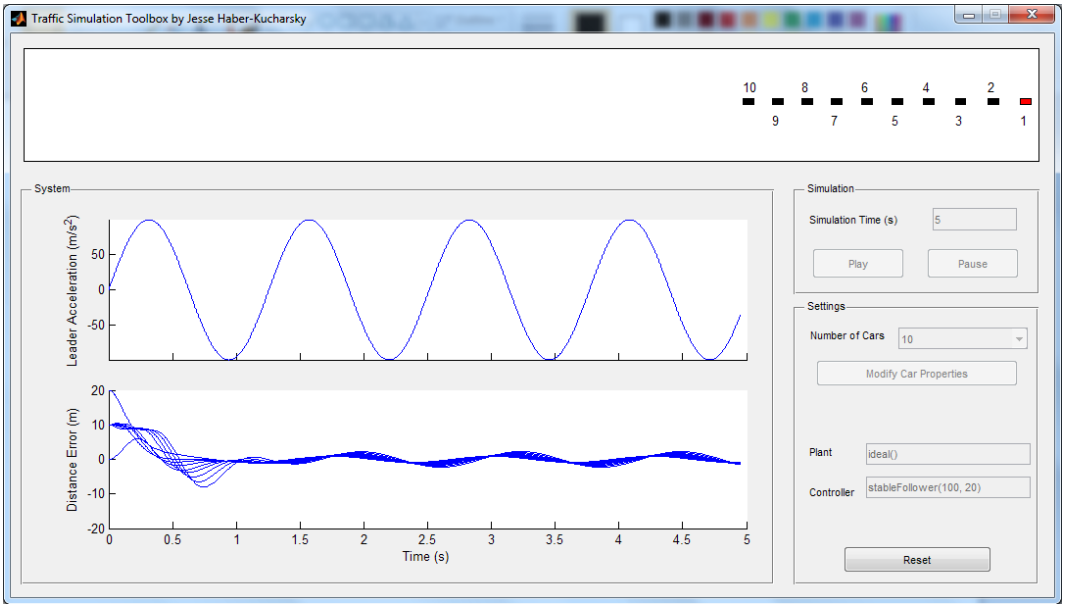

Figure 3: The main window of the Traffic Simulation Toolbox after the second simulation has completed.

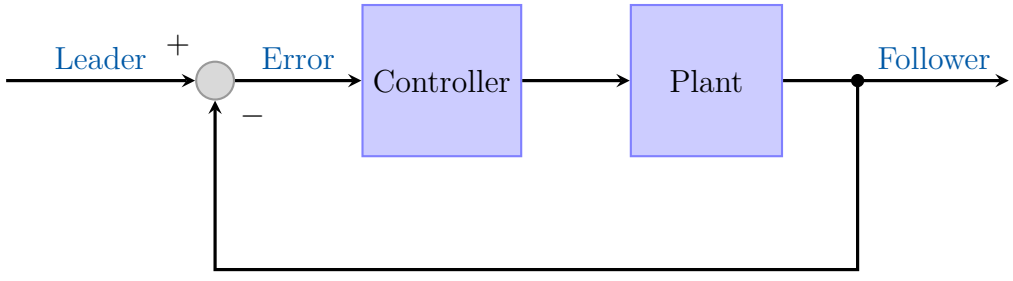

Figure 4: An overview of the feedback-control system model governing the behaviour of each car.

from its acceleration such that the velocity and position of the car at some instant are a function of the acceleration of that car until *just before* that instant. i.e. $v(t_i)$ and $p(t_i)$ – the car's velocity and position – are functions of its acceleration up to and including t_{i-1}.

Each car has associated with it some reference car (or as described in the quick-start guide, a leader). The motion of this leader car (it's acceleration, velocity, and position) serves as the ideal reference for the motion of the car. This leader reference is denoted as *Leader* in the figure.

The motion of the car is then compared in some way against the reference car and the result is some *error* signal which indicates the extent of the difference between the car's motion and its ideal motion. In many cases, this is simply the difference between, for example, the velocity of the leader and the velocity of the car.

The *Controller* calculates the new acceleration of the car as some function of the error signals. In the quick-start guide above, the controller consisted of the sum of the error in the distance and velocity with a gain of K_1 and K_2 respectively. i.e.,

$$a(t_i) = K_1 e_d(t_i) + K_2 e_v(t_i)$$

In reality, the Traffic Simulation Toolbox does not restrict the controller to be a function of the errors. The controller can be any arbitrary function of the car's motion and the motion of its leader. However, most controllers will result in unstable systems unless they are carefully chosen.

The *plant* represents the car itself and its behaviour in the absence of any external controller. Once the acceleration of the car has been determined by the controller, it can be further modified by the natural behaviour of the car itself. For example, if there is wind resistance present, then the acceleration of the car could be modelled as follows:

$$a_p(t_i) = a_c(t_i) - K v_c(t_i)$$

$a_c(t_i)$ and $v_c(t_i)$ represent the acceleration and velocity of the car as determined from the controller and $a_p(t_i)$ represents the new acceleration taking into account the effect of wind resistance which is here modelled as a constant gain of the car's velocity.

Finally, the resulting acceleration of the car is fed back and compared against the new behaviour of the leader and the process begins again. This

is the crucial step, without which stable systems would not be possible. The feedback compensates for any abrupt changes in the leader's motion.

For example, consider the simple controller described above. If the leader car begins to accelerate rapidly, then the differences between its velocity and position and the follower's velocity and position will increase rapidly as well. In other words, the errors will increase. These increased errors will result in an acceleration in the car which will itself reduce the difference in velocity and position. In conclusion, the system compensates for deviations in the car's motion as compared to the reference car.

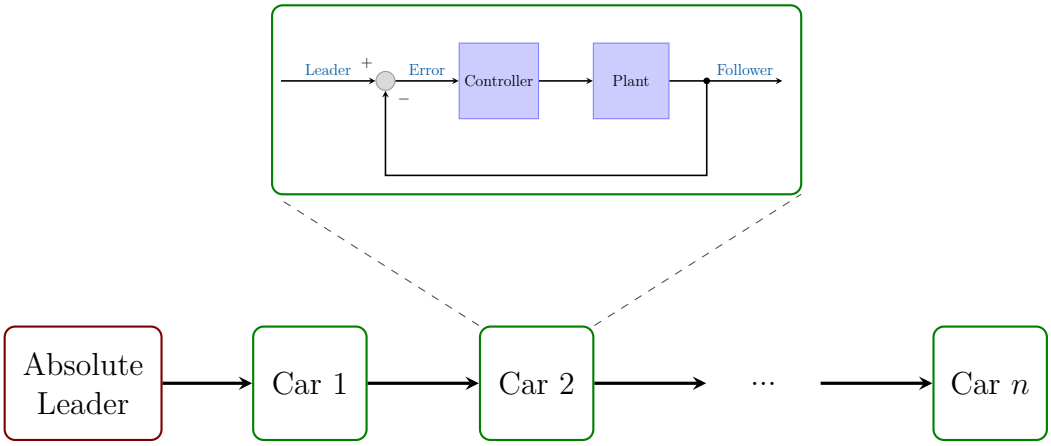

Figure 5: A sequence of cars led by the absolute leader where each follower car is governed by the control system shown in Figure 4

.

The Traffic Simulation Toolbox models a sequence of cars as a chain of individual cars where each is governed by the control system shown in Figure 4. Figure 5 shows the sequence. In the default case, each car's leader reference is the car directly ahead of it in the sequence. The sequence of cars is lead by a car whose motion is predetermined; its acceleration is given as some function of time. This car is named the *absolute leader* since its motion is not effected by any other cars.

2.3 The Main Window

Much of the functionality of the main window of the Traffic Simulation Toolbox is explained in the quick-start example of Section 2.1. However, this section will describe it in more depth.

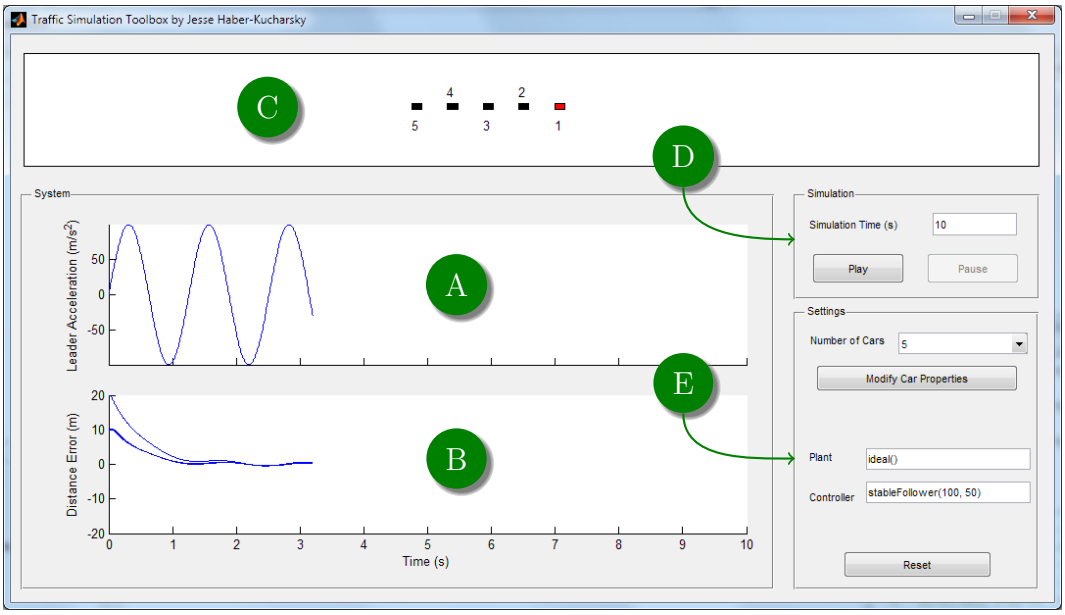

Figure 6: The main window of the Traffic Simulation Toolbox showing a simulation in-progress after it has been paused.

Figure 6 shows the main window of the Traffic Simulation Toolbox running a simulation. The different parts of the window are annotated and described below.

A

Leader Acceleration Plot

The motion of the absolute leader car – the car that leads the sequence – is specified exactly in terms of its acceleration as a function of time. This plot simply shows the acceleration of the leader car up-to-and-including the current instant of time in the simulation.

In the figure (and by default), the leader's acceleration is a sinusoidal function.

The exact function which gives the acceleration is entirely arbitrary and can be customized in the car properties dialogue, which is explained in Section 2.4.

B

Distance Error Plot

A natural indicator of the effectiveness of a given controller is to examine the magnitude of the difference between a car's ideal position and its actual position (i.e. the distance error).

This plot shows the distance error for multiple cars in the sequence. For performance reasons, the distance error is only shown for a *selection* of cars in the sequence. The choice and number of cars which have their distance error plotted depends on the number of cars in the sequence, as shown in Table 1.

C

Traffic Animation

While the simulation is running an animation is shown which includes the position and velocity of each car, here represented by a numbered black rectangle.

The leader car is distinguished in the animation by a red rectangle.

The left side of the animation area represents a position of 0 m. The position of the right side of the animation area is dynamic. In general, it will adjust itself so that at the end of the simulation the leader car will end up at the right side of the animation area.

D

Simulation Configuration

This section of the main window is used to adjust the simulation time (in seconds) and control the simulation once it has started. A running simulation can be paused and resumed an arbitrary number of times.

E

Settings

This section of the main window is used to adjust the properties of the cars in the sequence and the models used for the plant and the controller.

The number of cars is selected by using the drop-down menu. The minimum number of cars is 5, and the maximum (for performance and space reasons) is 35.

The button labelled `Modify Car Properties` is used to launch the car properties dialogue which is described in Section 2.4.

Finally, the models and parameters for the controller and plant are specified here. For more information on the models included with the Traffic Simulation Toolbox and how to use them, see Section 3 where the subject is covered exclusively.

Table 1: For performance reasons, the distance error is not plotted for every car in the sequence. The selection of cars for which it is plotted depends on the number of cars in the sequence as shown.

Cars in Sequence	Cars with Error Distance Shown	Selection Criterion
5	4	Every car
10	9	Every car
15	14	Every car
20	7	Every 3^{rd} car
25	8	Every 3^{rd} car
30	6	Every 5^{th} car

2.4 Adjusting Car Properties

The car properties dialogue is opened by pressing the button labelled `Modify Car Properties` as shown in Figure 6. The dialogue is shown in Figure 7.

Initial Conditions and Leader References

Each car in the sequence is identified with an integer. The absolute leader has the identifier of 1 and each subsequent car is identified by the next integer $(2, 3, 4, \ldots, n)$.

Before the simulation begins, each car in the sequence begins at some fixed position. Once the simulation begins, each car has some initial velocity.

Both of these parameters can be specified independently for each car in the sequence.

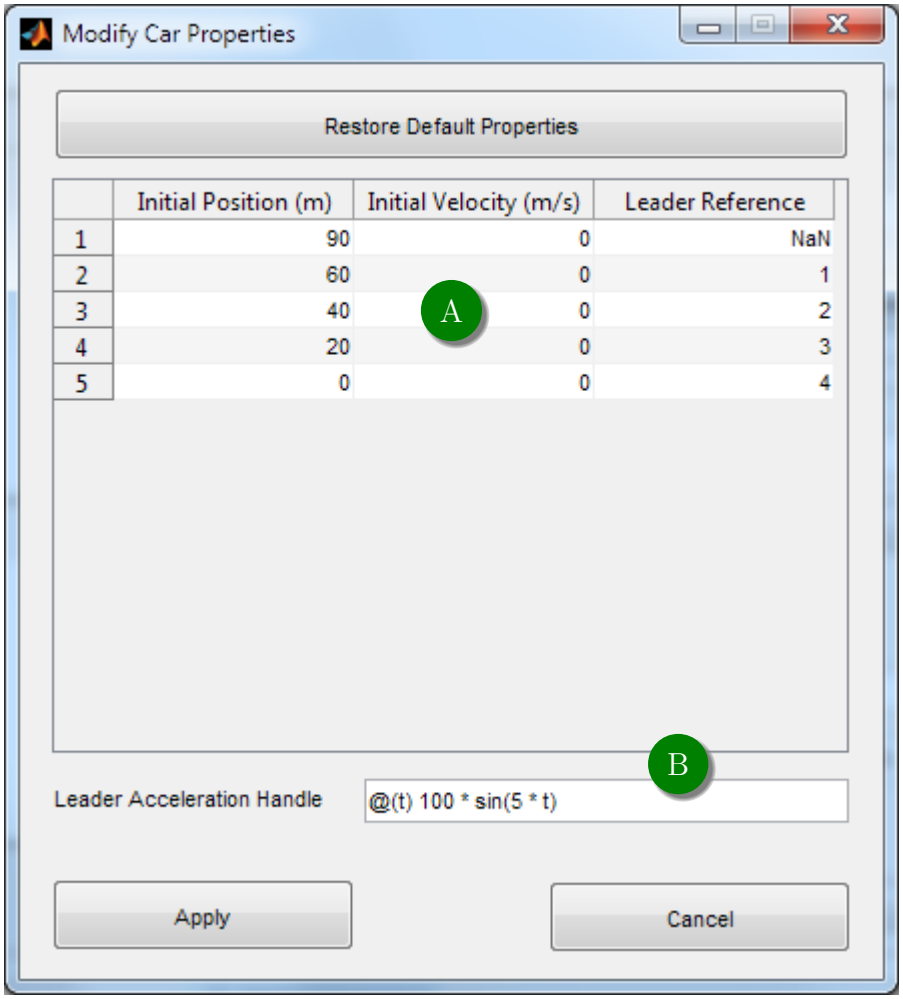

Figure 7: The default state of the car properties dialogue.

As mentioned many times in this manual, the default leader reference of each car in the sequence is the car immediately in front of it. As shown in the figure, the leader reference for car 2 is car 1, the leader for car 3 is car 2, and so on. A natural restriction on the leader reference of a car is that it must be a car which is ahead in the sequence. This means that the leader reference of car 3 could *not* be car 5.

In the car properties window, each car's properties are given in a row in the table. To change any of the properties for a car, simply edit the appropriate cell in the table. If an invalid value is entered for a given cell, then a context-appropriate error message will be displayed.

B **Leader Acceleration**

The motion of the absolute leader of the sequence of cars is given by its acceleration as a function of time.

Any valid MATLAB function handle can be used as the acceleration function, provided that the function itself obeys the following conditions:

1. The function must accept a vector of size N consisting of time values as input. Additional parameters are allowed, but the function *must accept an invocation with only the time parameter.*
2. The function must return as output a vector of size N consisting of the acceleration of the leader at each time value given as input.

For examples of different acceleration functions, see Table 2. For an in-depth description of the different types of function handles, see the MATLAB documentation.

The following section goes through the process of crafting an example acceleration function.

2.4.1 Example: Acceleration Function

This example will work through the process of crafting an appropriate acceleration function in order to model a specific scenario in the Traffic Simulation Toolbox.

Table 2: Examples of different leader acceleration functions.

Example	Description
`@sin`	The handle for the $\sin(t)$ function built into MATLAB.
`@(t) 5 * ones(size(t))`	A constant acceleration of $5\,\mathrm{m/s^2}$.
`@(t) foobar(t, 15)`	User-defined function with two parameters.
`@baz`	A reference to a user-defined m-file.
`@(t) abruptStop(t, 20, 3, 0.2)`	See Section 2.4.1.

We would like to model the impact on traffic if a car abruptly stops, and then resumes its previous speed. The car may have been trying to avoid running over a squirrel on the road.

The words "abruptly stop" could be interpreted to mean that the car – travelling at some velocity v_0 – suddenly stops at some time t_0. A short time later, the car resumes travelling at its previous velocity. This velocity function is plotted in Figure 8.

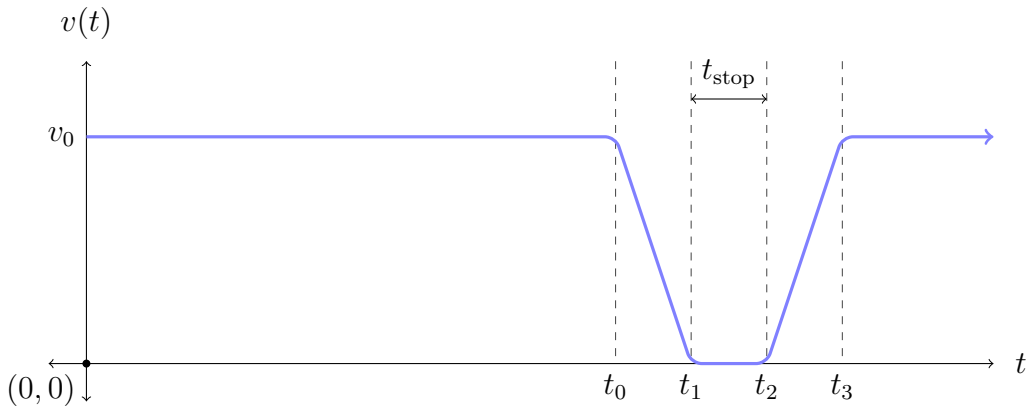

Figure 8: The desired velocity function of the leader car.

Since the motion of the leader car must be expressed in terms of its acceleration, we need to translate the above velocity function into its equivalent acceleration function. The equivalent acceleration function – the first derivative of the velocity function – is shown in Figure 9.

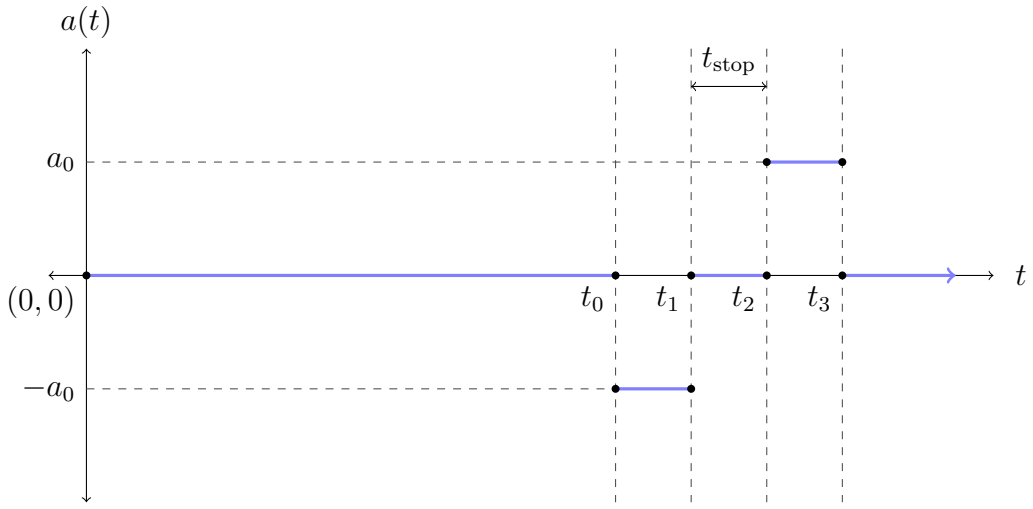

Figure 9: The equivalent acceleration function of the car with the velocity function shown in Figure 8.

We can see that in order to bring the car to a complete stop, we must have:

$$a_0 = \frac{v_0}{t_3 - t_2} = \frac{v_0}{t_1 - t_0}$$

assuming that $t_1 - t_0 = t_3 - t_2$.

Writing a function in MATLAB to produce this acceleration is fairly easy. The listing is shown in Figure 10.

Running the simulation while with the new function (and some example parameter values) results in the figure shown in Figure 11.

3 Using, Defining, and Modifying Models

The usefulness of the Traffic Simulation Toolbox comes from the flexibility allowed in defining models for the controller and the plant.

Each model calculates the motion of a follower car in a different matter. Models can also include accept parameters which further refine their behaviour.

```
%% abruptStop.m
%%
function a = abruptStop(t, initialVelocity, stopTime, stopDuration)

% t
% -- The time vector supplied automatically to the function.

% initialVelocity
% -- The initial velocity specified for the leader
%    car in the car properties table. This is used
%    to calculate the necessary acceleration
%    to bring the car to a complete stop.

% stopTime
% -- The instant in time (in seconds) at which the car
%    should begin braking.

% stopDuration
% -- The duration of time (in seconds) that the car should
%    stay stationary.
%%

a = zeros(size(t), 'double');

% We will assume a constant braking time.
brakingDuration = 0.2;
accel = initialVelocity / brakingDuration;

% The car should deaccelerate during the braking period.
a((t >= stopTime) & (t < stopTime + brakingDuration)) = -accel;

% The car should accelerate after the stop period.
a((t >= stopTime + brakingDuration + stopDuration) ...
    & (t < stopTime + stopDuration + 2 * brakingDuration)) = accel;
end
```

Figure 10: The code listing for abruptStop.m in Section 2.4.1.

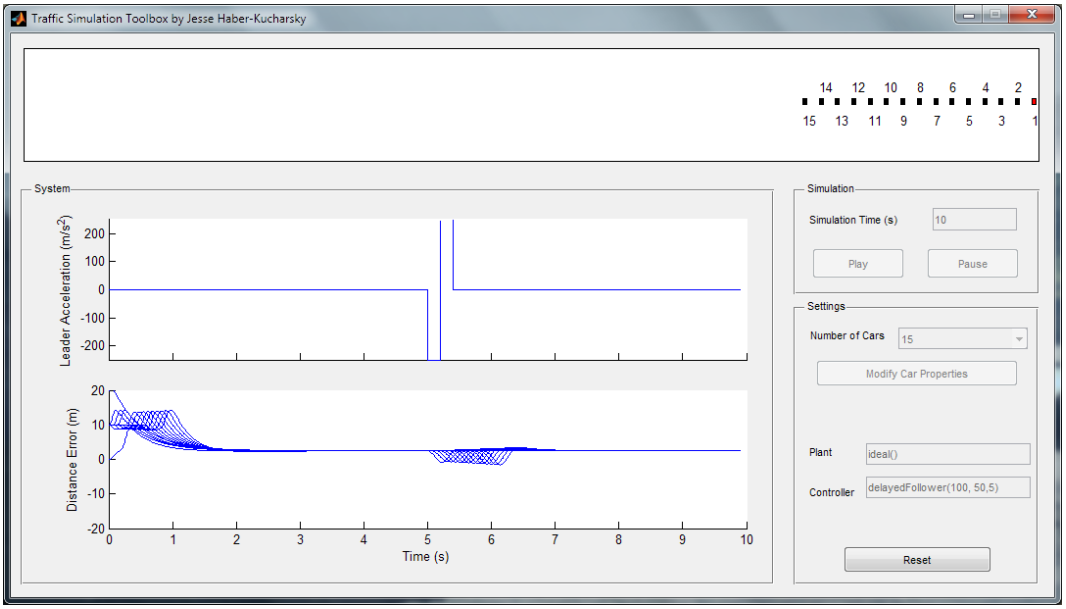

Figure 11: The main window after a simulation has completed where the acceleration of the leader is given by `abruptStop`.

Models and their parameters are specified using syntax similar to a function invocation. For example, a model `exampleController` with parameters A and B could be specified in the main window as `exampleController(23,-5)` with $A = 23$ and $B = -5$.

In the Traffic Simulation Toolbox distribution, models for controllers and plants are simply MATLAB m-files stored in the `+controllers` and `+plants` directories, respectively.

3.1 Included Models

Traffic Simulation Toolbox comes with only a small selection of models. They are explained below.

3.1.1 Controllers

`stableFollower(K1, K2)` The `stableFollower` controller (also described in the quick-start example of Section 2.1) determines the car's acceleration as a weighted sum of the distance error and the velocity error as shown:

$$a(t_i) = K_1 \left[p_L(t_i) - p(t_i) - D \right] + K_2 \left[v_L(t_i) - v(t_i) \right]$$

It can be shown through analysis in the frequency domain that this controller will only result in stable systems provided that the following conditions are met:

$$\begin{aligned} K_1 &> 0 \\ K_2 &> 2\sqrt{K_1} \end{aligned}$$

These conditions are derived in Appendix A.

`delayedFollower(K1, K2, T)` The `delayedFollower` controller is similar to the `stableFollower` controller except that it incorporates a time delay which accounts for the reaction time of the driver of the car. The T parameter is the number of "ticks" that elapse before the car responds to a change in the reference signal.

Mathematically, this controller could be expressed as

$$a(t_i) = \begin{cases} a_{\mathrm{SF}}(1) & \text{if } t_i \leq T \\ a_{\mathrm{SF}}(t_i - T) & \text{Otherwise} \end{cases}$$

where a_{SF} is the `stableFollower` controller previously discussed.

An important point is that internally, the number of discrete points is a constant. Therefore, as the simulation time is increased, the duration of a fixed number of "ticks" also increases.

3.1.2 Plants

`ideal()` This plant represents a system in which the motion of the plant (car) is strictly determined by the controller. This plant model has no effect on the acceleration of the car. In this way, it functions as a placeholder plant for situations in which the effect of disturbances are not being investigated as part of a simulation.

$$a_p(t_i) = a_c(t_i)$$

`windResistance(W)` This plant models the effects of a constant wind resistance which is proportional to the velocity of the car.

$$a_p(t_i) = -Wv(t_i) + a_c(t_i)$$

3.2 Defining and Modifying Models

The Traffic Simulation Toolbox stores the acceleration, velocity, and position of each car individually as arrays.

Every controller or plant model must return a MATLAB function handle to a function with exactly three parameters: l, f, k. These are all integer indices. l is the index of the car which is the leader reference for the car in question. f is the index of the car itself. Finally, k is the index of the current instant in time.

From these indices, the acceleration of the current time (at the k^{th} instant) can be calculated from the position and velocity of the car and its leader.

For an example of writing a model, it is advisable to look at the source files for models included with the Traffic Simulation Toolbox.

A Derivation of Conditions for `stableFollower`

As described in Section 3.1.1, the `stableFollower` controller calculates the acceleration at some instant as follows:

$$a(t) = K_1 \left[p_L(t) - p(t) - D\right] + K_2 \left[v_L(t) - v(t)\right]$$

The Traffic Simulation Toolbox stores acceleration, velocity, and position values as discrete values. For the purposes of analysis, we have written the equation of motion as if each signal were continuous.

We would like to determine the possible values of K_1 and K_2 such that the sequence of traffic is *stable*. By this, we mean that each car maintains its ideal separation distance.

We can define the positional error signal as follows:

$$e(t) = p_L(t) - p(t) - D$$

such that

$$\dot{e}(t) = \dot{p_L}(t) - \dot{p}(t)$$

and

$$\ddot{e}(t) = \ddot{p_L}(t) - \ddot{p}(t) \tag{1}$$

where the dot indicates the first derivative with respect to time (and each additional dot indicates the next derivative).

We can therefore re-express the acceleration of the car as a function of the error signals and their derivatives:

$$a(t) = K_1 e(t) + K_2 \dot{e}(t) \tag{2}$$

Note also that the acceleration of the car is simply the second derivative of its position:

$$a(t) = \ddot{p}(t)$$

Recognizing this fact and combining Equation (1) and Equation (2), we get:

$$\ddot{e}(t) = a_L(t) - K_1 e(t) - K_2 \dot{e}(t)$$

Rearranged into a more standard form, this relation is expressed as:

$$\ddot{e}(t) + K_2 \dot{e}(t) + K_1 e(t) = a_L(t) \tag{3}$$

Taking the Laplace transform of Equation (3) assuming zero-state conditions, we have:

$$s^2 E(s) + K_2 s E(s) + K_1 E(s) = A_L(s)$$

Finally, we can define the transfer function relating the position error to the leader acceleration:

$$H(s) = \frac{E(s)}{A_L(s)} = \frac{1}{s^2 + K_2 s + K_1} \tag{4}$$

This is a second-order system and can be analyzed using standard tools such as root-locus or Routh-Hurwitz. However, to make this discussion self-contained, we will provide a basic derivation of the range of allowable constants K_1 and K_2 here.

To analyze the stability of the system characterized by this transfer function, we can express it in pole-zero form:

$$H(s) = \frac{1}{(s-\alpha)(s-\beta)}$$

α and β are the poles of the system. Taking the inverse Laplace transform of the transfer function, we can obtain the impulse response:

$$h(t) = C_1 e^{\alpha t} + C_2 e^{\beta t}$$

where C_1 and C_2 are some yet undetermined constants and the characteristic modes are $e^{\alpha t}$ and $e^{\beta t}$.

To ensure asymptotic stability (and therefore BIBO stability), we need to ensure that the real part of both poles are negative (i.e., in the left part of the s-plane). However, we also wish to avoid oscillations. Therefore, we will choose the poles such that the system is overdamped and we require that both poles be negative real numbers.

The poles can be solved quadratically. They are:

$$\alpha = \frac{-K_2 + \sqrt{K_2^2 - 4K_1}}{2}$$
$$\beta = \frac{-K_2 - \sqrt{K_2^2 - 4K_1}}{2}$$

To satisfy the conditions that the poles both be negative real numbers, we can produce the following system of inequalities:

$$K_2^2 - 4K_1 \geq 0 \tag{5}$$

$$-K_2 + \sqrt{K_2^2 - 4K_1} < 0 \tag{6}$$

$$-K_2 - \sqrt{K_2^2 - 4K_1} < 0 \tag{7}$$

To begin, consider Equation (5). We have:

$$K_2{}^2 - 4K_1 \geq 0$$
$$K_2{}^2 \geq 4K_1$$

This implies that $K_2 \geq 2\sqrt{K_1}$ (a positive number) or $K_2 \leq -2\sqrt{K_1}$ (a negative number). If K_2 were negative, then Equation (6) could not be satisfied since both terms would be positive. Therefore, only the positive solution for K_2 is valid and

$$K_2 \geq 2\sqrt{K_1}$$

Now consider K_1. If K_1 were negative then $K_2 < \sqrt{K_2^2 - 4K_1}$ and Equation (6) could not be satisfied since we would have $-K_2 > -\sqrt{K_2^2 - 4K_1}$.

Therefore, we conclude that

$$K_1 > 0$$

Finally, note that if $K_1 > 0$ we have $K_2 > \sqrt{K_2^2 - 4K_1}$ and this is a re-statement of Equation (7).

Therefore, the constraints on K_1 and K_2 which result in a stable, overdamped system are:

$$K_1 > 0$$
$$K_2 \geq 2\sqrt{K_1}$$

B License

The source code distributed with the Traffic Simulation Toolbox is released under the terms of the following licence: