Suggested Changes to followerStopper and smoothUpParams

Chris Kreienkamp

July 2, 2019

1 Description

followerStopper and smoothUpParams are both MATLAB functions used in conjunction in Simulink to control the velocity of an autonomous vehicle. In order to dissipate traffic waves, a vehicle wants to drive at an optimal, or reference, velocity. If the car follows the vehicle directly ahead without following this reference velocity, the car will eventually have to brake hard in order to prevent a crash, and hard braking increases traffic. Sometimes, the car will have to slow down because following the reference velocity would result in a crash. followerStopper is a controller that will tell the car whether it can drive at the reference velocity or if it must drive at some fraction of that velocity.

followerStopper takes in the relative velocity between the lead vehicle and autonomous vehicle (AV), the relative distance between the lead vehicle and the autonomous vehicle, and the velocity of the autonomous vehicle to output a command velocity for the AV. followerStopper is a function of r (reference velocity), dx (relative position), dv (relative velocity), v_AV (autonomous vehicle velocity), dx_min (minimum distance between lead vehicle and AV), dx_activate (distance when AV should travel slower than the reference velocity).

smoothUpParams edits the reference, or optimal, velocity before it gets to followerStopper in order to prevent harsh and unnecessary acceleration or deceleration. smoothUpParams is a function of the max_speed (reference velocity), vel (autonomous vehicle velocity), max_accel (maximum comfortable acceleration for the autonomous vehicle), and max_decel (maximum comfortable deceleration for the autonomous vehicle. Using these inputs, smoothUpParams edits the reference velocity so that it is a reasonable value when it is sent to the followerStopper controller. Suppose, for example, the speed limit were to change from 10 m/s to 15 m/s and no car was in front of the AV. followerStopper would command the AV to travel at the reference velocity of 15 m/s immediately, but this will result in a large acceleration and an overshoot. smoothUpParams will cause the reference velocity to increase at a slow rate so that followerStopper can still command the reference velocity but it will not result in a large acceleration.

2 Problem Statement

A Simulink model was created with two cars. The lead car, initially positioned at 10 m in front of the follower car, was given a synthetic pre-recorded velocity from an actual human driver. The follower car, modeled as an autonomous vehicle, was controlled by smoothUpParams and followerStopper. As seen in Figure 1, there were sharp fluctuations in the reference velocity and the command velocity during the first 5 s, and there were sharp fluctuations in the command velocity around 78 s. Such sharp fluctuations resulted in an unacceptably large acceleration. The goal of this document is to present edits to the MATLAB smoothUpParams and followerStopper codes that will eliminate these sharp fluctuations for this specific lead vehicle velocity profile and for all lead profiles.

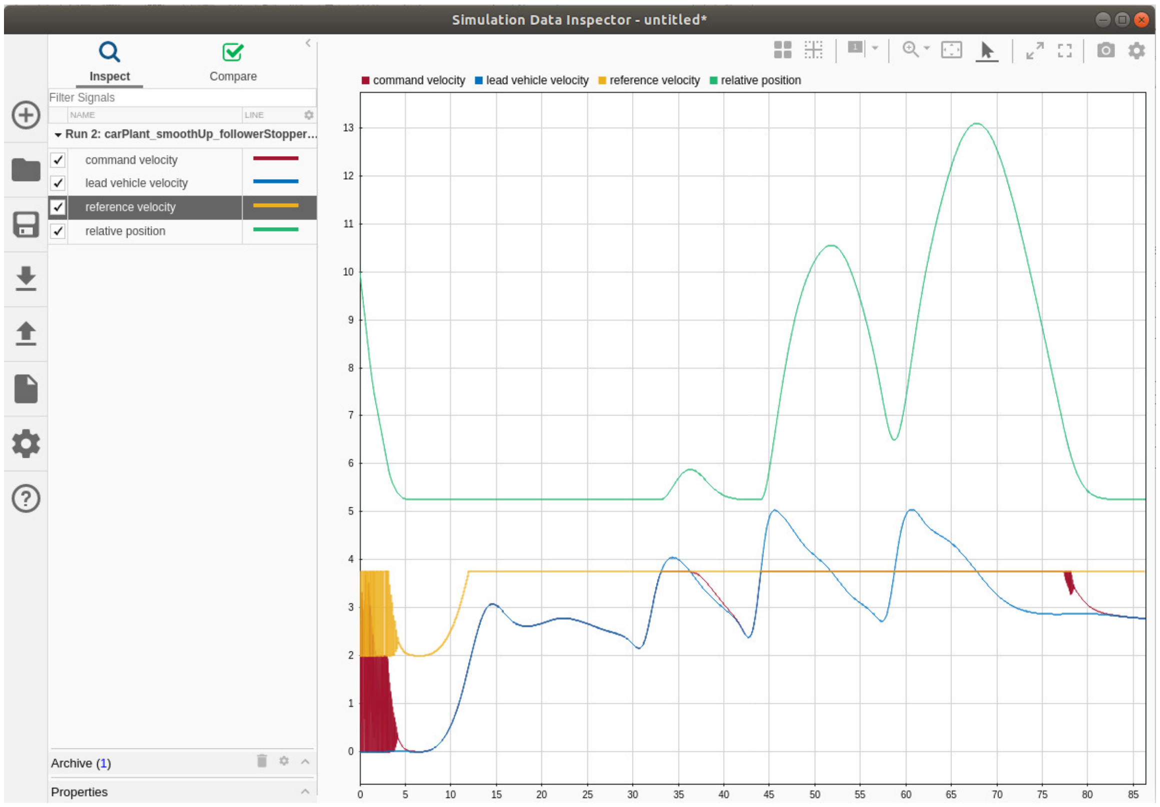
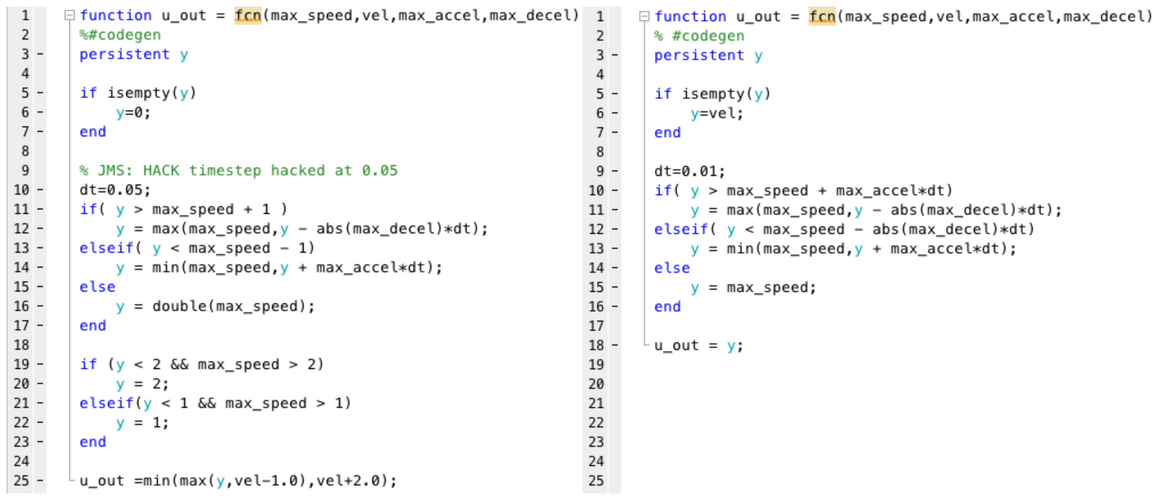


Figure 1: Original followerStopper and original smoothUpParams.

3 Editing smoothUpParams

3.1 Original code

The original code is displayed on the left side of Figure 2. The function relies on a persistent variable `y` to remember what the reference velocity either was or was near, so that if the reference velocity is changed, the code will know it needs to incrementally move toward the new reference velocity. `y` can be treated as the output reference velocity from the previous time-step. If there was no previous value of `y`, it will be set to 0. The code assumes a time-step of 0.05 s, and then proceeds through an `if` statement. The `if` statement determines if `y` is higher than, less than, or within 1 of the `max_speed`, which is the reference velocity provided as an input. If `y` is higher than the `max_speed` + 1, then `y` will decrease at a deceleration determined by `max_decel`. If `y` is lower than the `max_speed` − 1, then `y` will decrease at a deceleration determined by `max_decel`. Otherwise, `y` will be set equal to `max_speed`. After exiting the `if` statement, the code enters another, checking if `y` is less than 1 or 2 and setting it equal to those respective values if so. The rationale is that reference velocity should at least be greater than 0 because followerStopper needs the reference velocity to be a nonzero value in order to give the car a nonzero velocity. If the reference velocity is too fast, followerStopper will protect the AV. The code ends by assuring that `y` is at least 1 or 2 and is not less than the current AV velocity by more than 1 m/s.



```
1   function u_out = fcn(max_speed,vel,max_accel,max_decel)
2   %#codegen
3   persistent y
4
5   if isempty(y)
6       y=0;
7   end
8
9   % JMS: HACK timestep hacked at 0.05
10  dt=0.05;
11  if( y > max_speed + 1 )
12      y = max(max_speed,y - abs(max_decel)*dt);
13  elseif( y < max_speed - 1)
14      y = min(max_speed,y + max_accel*dt);
15  else
16      y = double(max_speed);
17  end
18
19  if (y < 2 && max_speed > 2)
20      y = 2;
21  elseif(y < 1 && max_speed > 1)
22      y = 1;
23  end
24
25  u_out =min(max(y,vel-1.0),vel+2.0);
```

```
1   function u_out = fcn(max_speed,vel,max_accel,max_decel)
2   % #codegen
3   persistent y
4
5   if isempty(y)
6       y=vel;
7   end
8
9   dt=0.01;
10  if( y > max_speed + max_accel*dt)
11      y = max(max_speed,y - abs(max_decel)*dt);
12  elseif( y < max_speed - abs(max_decel)*dt)
13      y = min(max_speed,y + max_accel*dt);
14  else
15      y = max_speed;
16  end
17
18  u_out = y;
19
20
21
22
23
24
25
```

Figure 2: Original smoothUpParams code (left). Edited smoothUpParams code (right).

3.2 Edited code

The changes to the original code are described in detail below. Figure 3 displays the new data with the same followerStopper controller but with the edited smoothUpParams controller. As expected, the edits eliminate all sharp oscillations in the reference velocity in yellow. Consequently, the command velocity in red also has less oscillations at the very beginning of the experiment from about 0-1 s.

3.2.1 Initial value of `y`

The edited code is displayed on the right side of Figure 2. Focus will be given to the edits. If there was no previous value of `y`, it will be set to the current velocity of the autonomous vehicle instead of 0. Consider if the AV is not using followerStopper and then initiates followerStopper when already in motion, perhaps on a ramp to get onto a highway. If there are no cars ahead, followerStopper will tell the AV to follow the reference velocity. If `y` is set to the velocity of AV, then a new instantaneous reference velocity will be treated no differently than how the AV would normally react to a new input reference velocity. If $\texttt{y} = 0$, then from line 25 of the original code, the reference velocity $\texttt{u\_out} = \texttt{vel} - 1.0$ until $\texttt{y} > \texttt{vel} - 1.0$, which could take a while if the AV velocity is high because `y` can only increment by velocity values that respect the `max_accel` value.

3.2.2 Time step

The time-step is changed from $\texttt{dt} = 0.05$ to $\texttt{dt} = 0.01$ because the Simulink model has a time-step of 0.01 s.

3.2.3 `if` statements

Suppose the AV is traveling at the reference velocity of 5 m/s when it instantaneously changes to 5.9 m/s, but no cars are in front of it. Proceeding through the original code, the code would enter the `else` block, then `y` $= 5.9$. Proceeding to the end, `u_out` $= 5.9$. followerStopper would then command 5.9 m/s as the velocity, but this would be clearly wrong as changing from 5.0 m/s to 5.9 m/s over the course of 0.01 s would be an acceleration of 90 $\frac{m}{s^2}$! Instead, `y`*shouldhaveenteredthe*`elseif` block. Assuming `dt` $=$ `0.01` and `max_accel` $=$ `1.5`, then `y` $=$ `5.15`, a much more reasonable value. Therefore, the `if` statement is changed so that `y` will enter the `if` block or the `elseif` block if it is more than `max_accel` $*$ `dt` or `max_decel` $*$ `dt` away from the newly input reference velocity `max_speed`.

In terms of the second `if` statement, it is undesirable to have the reference velocity set to 1 m/s or 2 m/s if the car is not moving. In this particular example, the lead vehicle started 10 m ahead of the AV. followerStopper determined this was a safe distance, so commanded the AV to travel at the reference velocity. Therefore, the AV had to go from 0 m/s to 2 m/s over the course of 0.01 s, resulting in an acceleration of $200\frac{m}{s^2}$!

3.2.4 `u_out`

As described above, it is undesirable to use any reference velocity other than one that is spaced away from the previous reference velocity by an acceleration multiplied by a time-step. `y` was already computed to serve as this value, so it can be directly used for the new reference velocity `u_out`.

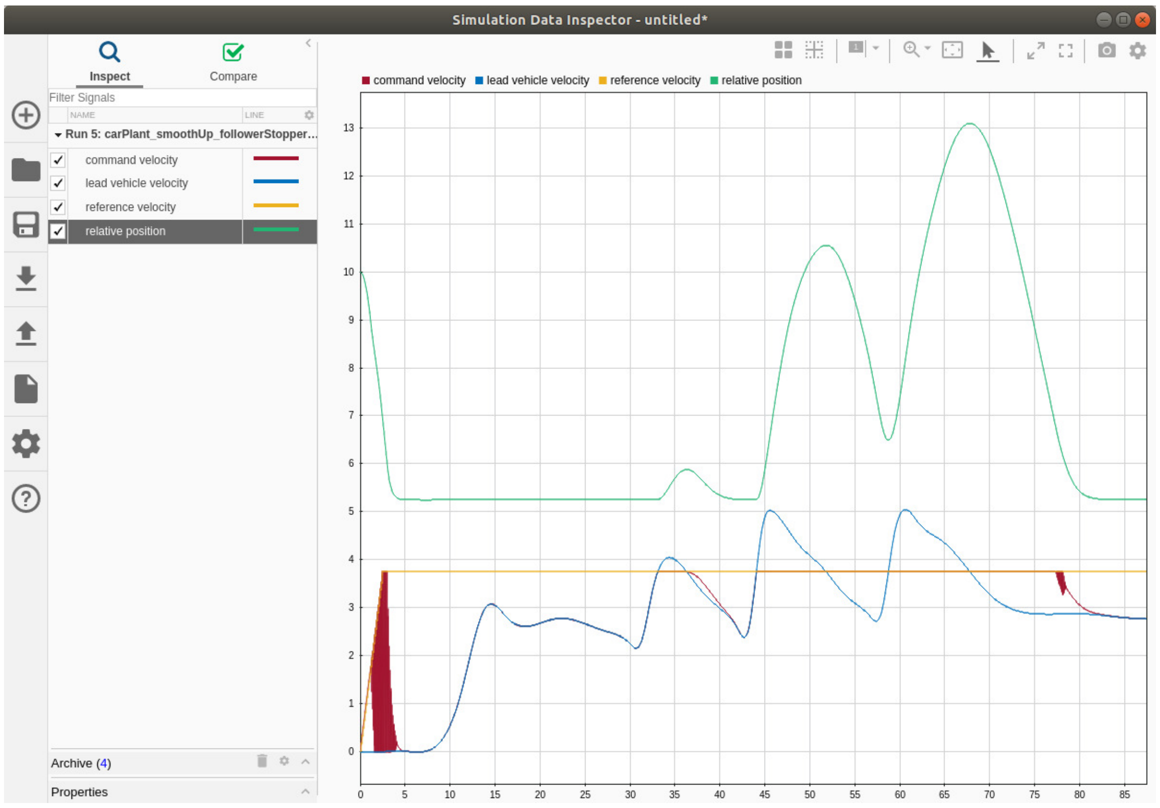


Figure 3: Original followerStopper and edited smoothUpParams.

4 Editing followerStopper

The changes to the original code are described in detail below. Figure 5 displays the new data with the edited (part 1) followerStopper controller and with the edited smoothUpParams controller. Though the reference velocity remains smooth, the commanded velocity in red is only slightly improved. As seen in the first few seconds, comparing the plot in Figure 5 to the previous plot in Figure 3 reveals that the oscillations at the very beginning are less dense, yet they occur for a longer time period. The oscillations at around 78 s did not change. Figure 7 displays the new data with the edited (part 2) followerStopper controller and with the edited smoothUpParams controller. In this figure, the acceleration of the AV is also added in light green to show that the acceleration never goes above 1.5 $\frac{m}{s^2}$. There are still small oscillations, but they are dramatically reduced from previous versions of smoothUpParams and followerStopper and are acceptable.

4.1 Original code

The original code is displayed on the left side of Figure 4, and the edited (part 1) code is on the right. The code, except for the last `if` statement, exactly follows the equations for FollowerStopper as laid out in "Real-Time Distance Estimation and Filtering of Vehicle Headways for Smoothing of Traffic Waves." That is,

$$x = x_{lead} - x_{AV} - L \quad (1)$$

$$\dot{x} = v_{lead} - v_{AV} \quad (2)$$

$$\xi_j(\dot{x}) = \omega_j + \frac{1}{2\alpha_j}(\dot{x}^*)^2 \text{ for j} = 1,2,3 \quad (3)$$

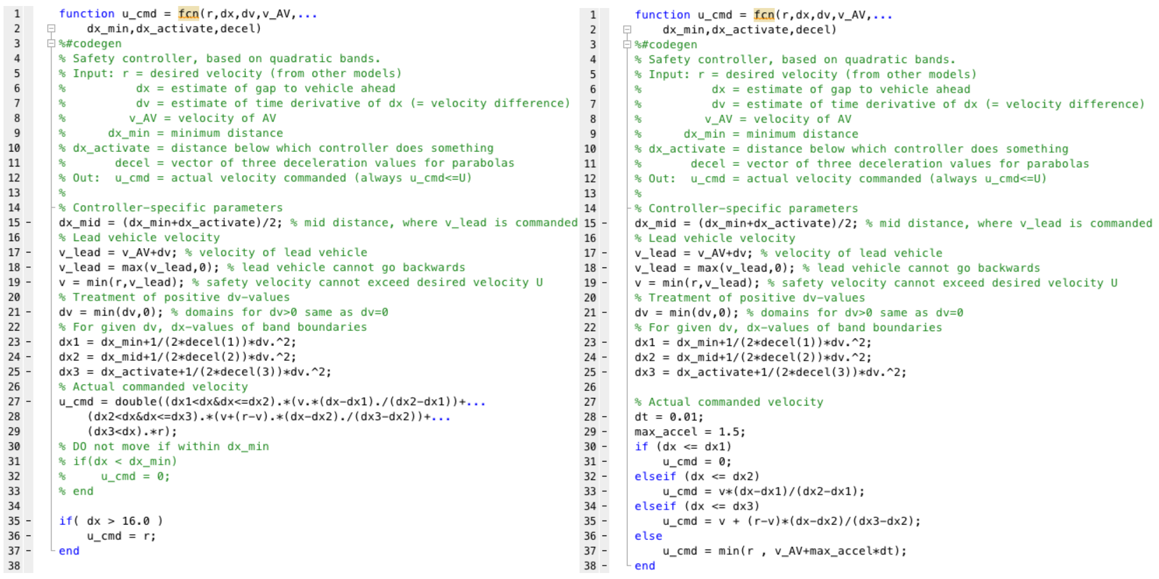
$$u = \begin{cases} 0 & x \leq \xi_1 \\ v\dfrac{x-\xi_1}{\xi_2-\xi_1} & \xi_1 \leq x \leq \xi_2 \\ v+(r-v)\dfrac{x-\xi_2}{\xi_3-\xi_2} & \xi_2 \leq x \leq \xi_3 \\ r & \xi_3 \leq x \end{cases} \quad (4)$$

where $\dot{x}^* = min(\dot{x}, 0)$. r is the reference velocity as taken from the output of smoothUpParams, u is the command velocity that is sent to the AV, and α_j and ω_j are parameters.

4.2 Edited code part 1

4.2.1 `if` statement

The original code is easy to follow until line 27. In that line, the original code does not account for the possibility `dx` $<=$ `dx1`. A similar concept is employed on line 32, but it is commented out and it is not entirely accurate because it compares `dx` to `dx_min`, not `dx1` as outlined in equation 4. Additionally, in line 35, the value `16.0` seems to be arbitrarily chosen when in reality `u_cmd` should equal the reference velocity only when `dx3` $<$ `dx`. The `if` statement follows equation 4 exactly. The only variation is line 37 in the edited code (part 1) in Figure 4 when



```
function u_cmd = fcn(r,dx,dv,v_AV,...
    dx_min,dx_activate,decel)
%#codegen
% Safety controller, based on quadratic bands.
% Input: r = desired velocity (from other models)
%          dx = estimate of gap to vehicle ahead
%          dv = estimate of time derivative of dx (= velocity difference)
%         v_AV = velocity of AV
%      dx_min = minimum distance
% dx_activate = distance below which controller does something
%       decel = vector of three deceleration values for parabolas
% Out:  u_cmd = actual velocity commanded (always u_cmd<=U)
%
% Controller-specific parameters
dx_mid = (dx_min+dx_activate)/2; % mid distance, where v_lead is commanded
% Lead vehicle velocity
v_lead = v_AV+dv; % velocity of lead vehicle
v_lead = max(v_lead,0); % lead vehicle cannot go backwards
v = min(r,v_lead); % safety velocity cannot exceed desired velocity U
% Treatment of positive dv-values
dv = min(dv,0); % domains for dv>0 same as dv=0
% For given dv, dx-values of band boundaries
dx1 = dx_min+1/(2*decel(1))*dv.^2;
dx2 = dx_mid+1/(2*decel(2))*dv.^2;
dx3 = dx_activate+1/(2*decel(3))*dv.^2;
% Actual commanded velocity
u_cmd = double((dx1<dx&dx<=dx2).*(v.*(dx-dx1)./(dx2-dx1))+...
    (dx2<dx&dx<=dx3).*(v+(r-v).*(dx-dx2)./(dx3-dx2))+...
    (dx3<dx).*r);
% DO not move if within dx_min
% if(dx < dx_min)
%     u_cmd = 0;
% end

if( dx > 16.0 )
    u_cmd = r;
end

```

```
function u_cmd = fcn(r,dx,dv,v_AV,...
    dx_min,dx_activate,decel)
%#codegen
% Safety controller, based on quadratic bands.
% Input: r = desired velocity (from other models)
%          dx = estimate of gap to vehicle ahead
%          dv = estimate of time derivative of dx (= velocity difference)
%         v_AV = velocity of AV
%      dx_min = minimum distance
% dx_activate = distance below which controller does something
%       decel = vector of three deceleration values for parabolas
% Out:  u_cmd = actual velocity commanded (always u_cmd<=U)
%
% Controller-specific parameters
dx_mid = (dx_min+dx_activate)/2; % mid distance, where v_lead is commanded
% Lead vehicle velocity
v_lead = v_AV+dv; % velocity of lead vehicle
v_lead = max(v_lead,0); % lead vehicle cannot go backwards
v = min(r,v_lead); % safety velocity cannot exceed desired velocity U
% Treatment of positive dv-values
dv = min(dv,0); % domains for dv>0 same as dv=0
% For given dv, dx-values of band boundaries
dx1 = dx_min+1/(2*decel(1))*dv.^2;
dx2 = dx_mid+1/(2*decel(2))*dv.^2;
dx3 = dx_activate+1/(2*decel(3))*dv.^2;

% Actual commanded velocity
dt = 0.01;
max_accel = 1.5;
if (dx <= dx1)
    u_cmd = 0;
elseif (dx <= dx2)
    u_cmd = v*(dx-dx1)/(dx2-dx1);
elseif (dx <= dx3)
    u_cmd = v + (r-v)*(dx-dx2)/(dx3-dx2);
else
    u_cmd = min(r , v_AV+max_accel*dt);
end
```

Figure 4: Original followerStopper code (left). Edited (part 1) followerStopper code (right).

$\mathtt{u\_cmd} = \mathtt{min}(\mathtt{r}, \mathtt{v\_AV} + \mathtt{max\_accel} * \mathtt{dt})$ because the car should not accelerate faster than its maximum desired acceleration to get to the reference velocity.

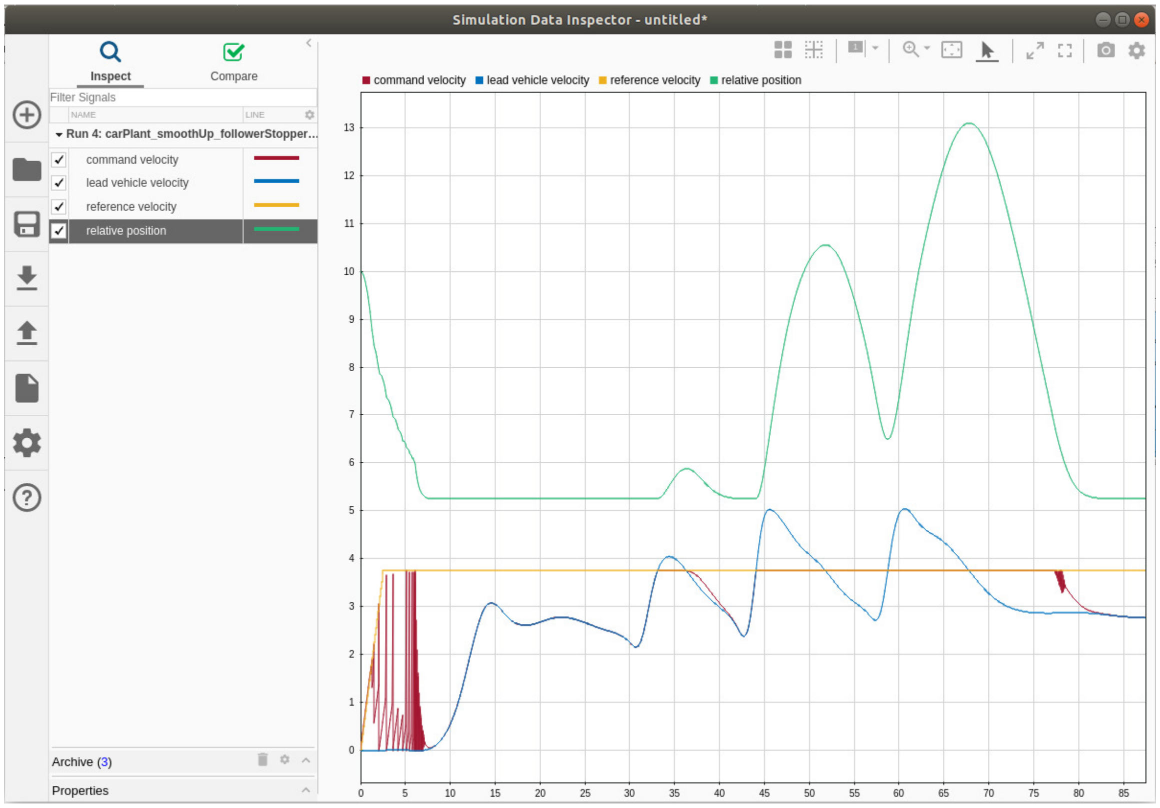


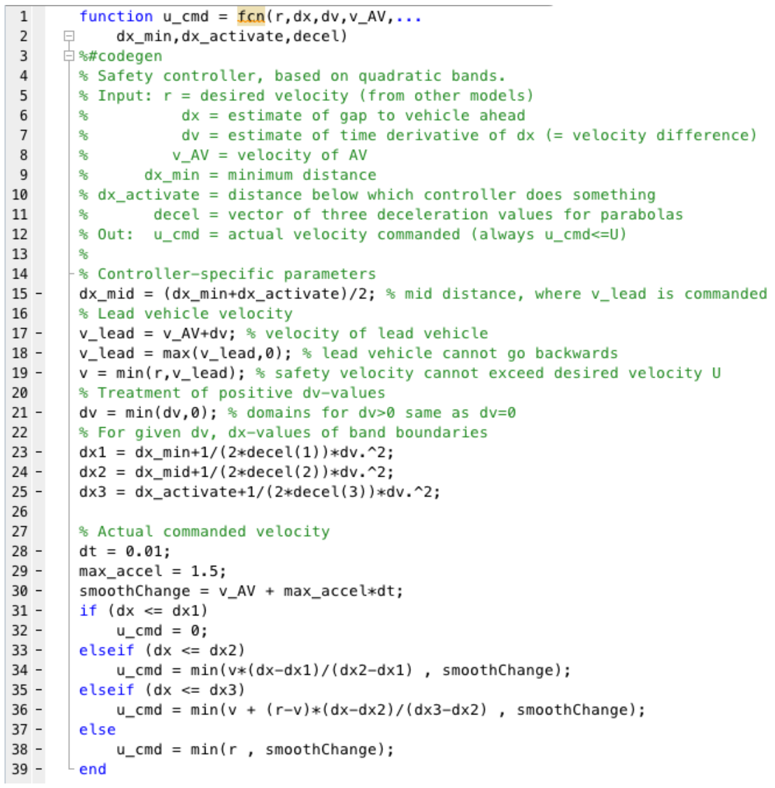
Figure 5: Edited (part 1) followerStopper and edited smoothUpParams.

4.3 Edited code part 2

Figure 6 displays the edited (part 2) code.

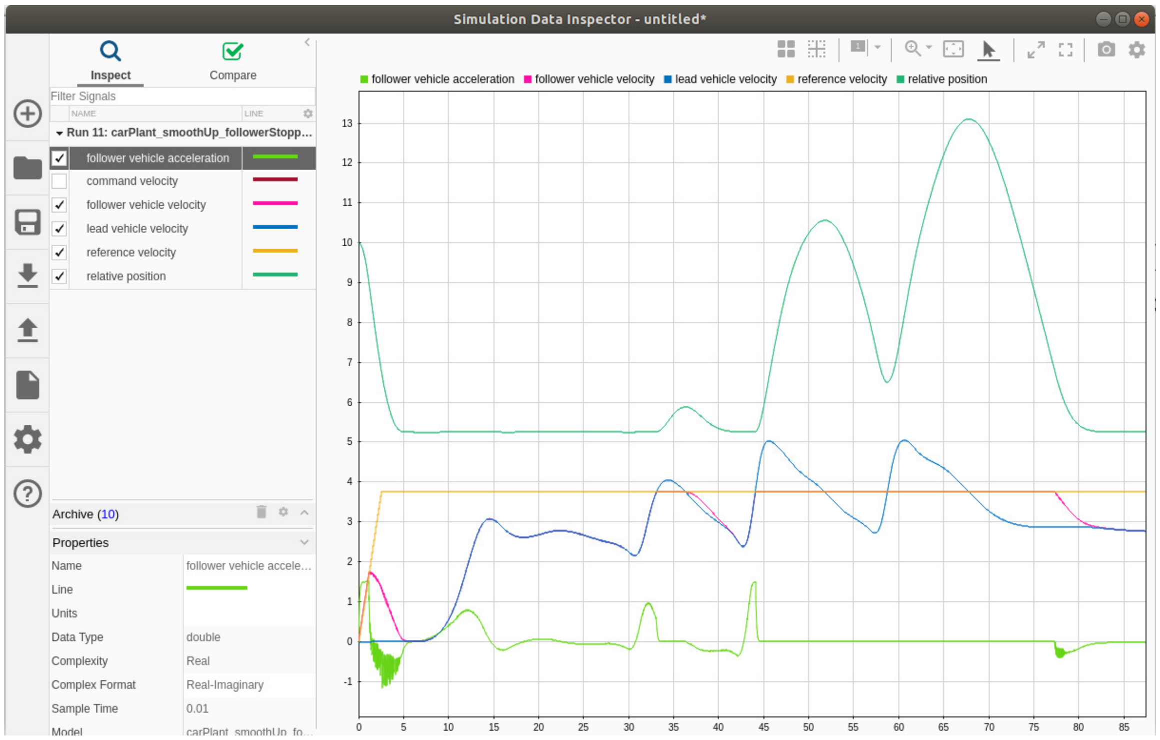
4.3.1 `smoothChange`

After spending some time looking at the oscillations, it was revealed that the sharp jumps occur when the relative distance `dx` crosses into a different piecewise area of the FollowerStopper in equation 4. If it was traveling at the reference velocity, the followerStopper edited code (part 1) made sure that it did not immediately travel at the reference velocity, but would incrementally work up to it. At points the AV would cross the `dx3` barrier, shooting up because that piecewise section calculated the command velocity `u` to be a large fraction of the much higher reference velocity. To solve this problem, a `smoothChange` variable, designed on line 30 in Figure 6, assured that no matter in what piecewise area that FollowerStopper was operating, it would not accelerate faster than the maximum acceleration. Though this could be tuned to deceleration as well, it is not desirable because FollowerStopper should not be limited in how quickly it can decelerate in case it needs to brake at full force to prevent a crash. Acceleration, however, is entirely dependent on rider comfort.



```
1   function u_cmd = fcn(r,dx,dv,v_AV,...
2       dx_min,dx_activate,decel)
3   %#codegen
4   % Safety controller, based on quadratic bands.
5   % Input: r = desired velocity (from other models)
6   %           dx = estimate of gap to vehicle ahead
7   %           dv = estimate of time derivative of dx (= velocity difference)
8   %          v_AV = velocity of AV
9   %       dx_min = minimum distance
10  % dx_activate = distance below which controller does something
11  %        decel = vector of three deceleration values for parabolas
12  % Out:  u_cmd = actual velocity commanded (always u_cmd<=U)
13  %
14  % Controller-specific parameters
15  dx_mid = (dx_min+dx_activate)/2; % mid distance, where v_lead is commanded
16  % Lead vehicle velocity
17  v_lead = v_AV+dv; % velocity of lead vehicle
18  v_lead = max(v_lead,0); % lead vehicle cannot go backwards
19  v = min(r,v_lead); % safety velocity cannot exceed desired velocity U
20  % Treatment of positive dv-values
21  dv = min(dv,0); % domains for dv>0 same as dv=0
22  % For given dv, dx-values of band boundaries
23  dx1 = dx_min+1/(2*decel(1))*dv.^2;
24  dx2 = dx_mid+1/(2*decel(2))*dv.^2;
25  dx3 = dx_activate+1/(2*decel(3))*dv.^2;
26
27  % Actual commanded velocity
28  dt = 0.01;
29  max_accel = 1.5;
30  smoothChange = v_AV + max_accel*dt;
31  if (dx <= dx1)
32      u_cmd = 0;
33  elseif (dx <= dx2)
34      u_cmd = min(v*(dx-dx1)/(dx2-dx1) , smoothChange);
35  elseif (dx <= dx3)
36      u_cmd = min(v + (r-v)*(dx-dx2)/(dx3-dx2) , smoothChange);
37  else
38      u_cmd = min(r , smoothChange);
39  end
```

Figure 6: Edited (part 2) followerStopper code.



Run 11: carPlant_smoothUp_followerStopp...

✓	follower vehicle acceleration
	command velocity
✓	follower vehicle velocity
✓	lead vehicle velocity
✓	reference velocity
✓	relative position

Name	follower vehicle accele...
Line	
Units	
Data Type	double
Complexity	Real
Complex Format	Real-Imaginary
Sample Time	0.01
Model	carPlant_smoothUp_fo...

Figure 7: Edited (part 2) followerStopper and edited smoothUpParams.

5 Appendix

Below are more figures for reference.

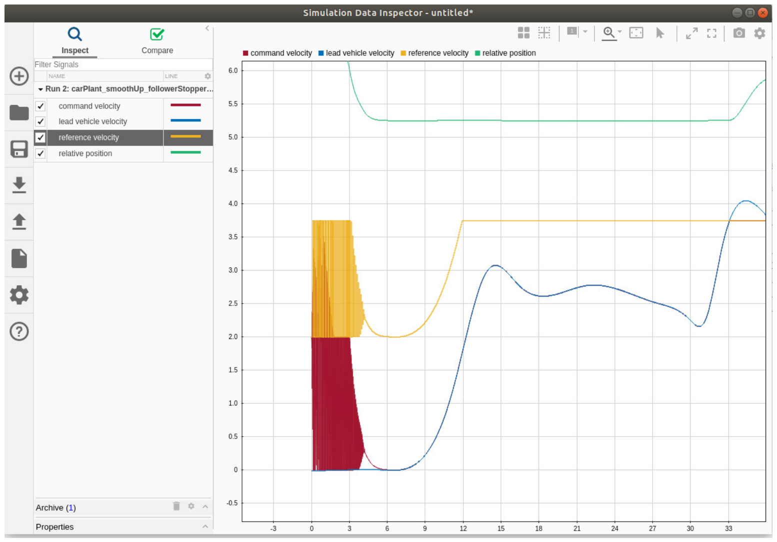


Figure 8: Original followerStopper and original smoothUpParams at the first oscillation set.

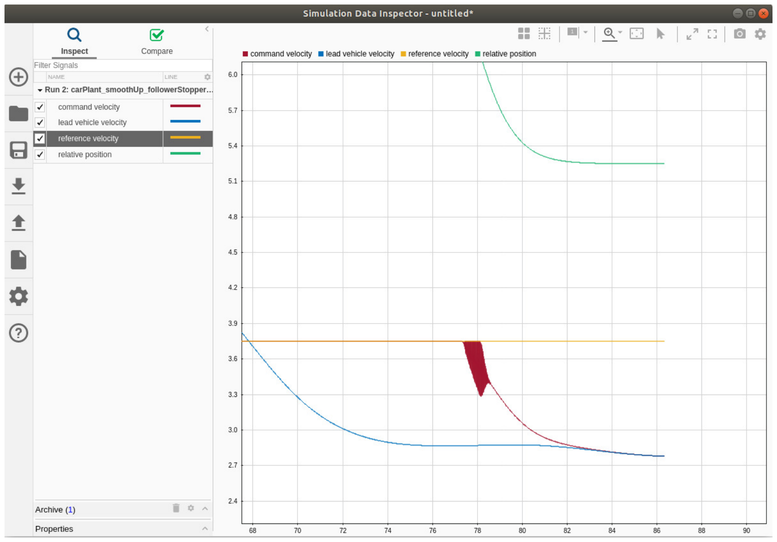


Figure 9: Original followerStopper and original smoothUpParams at the second oscillation set.

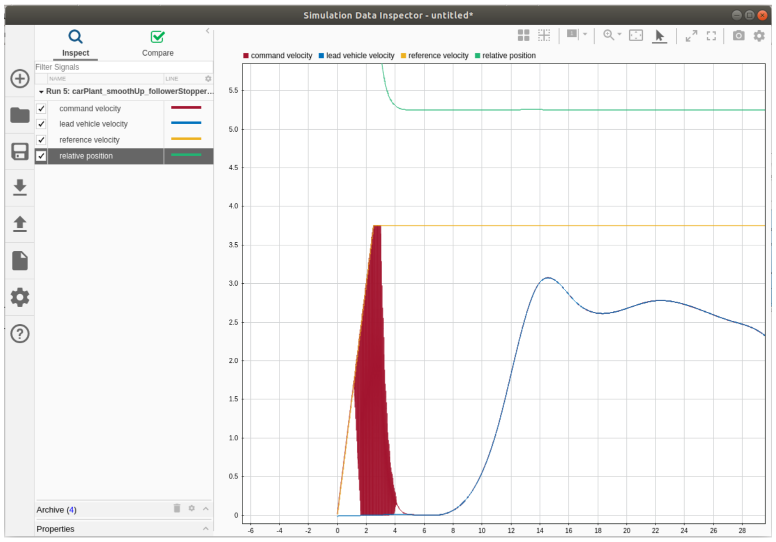


Figure 10: Original followerStopper and edited smoothUpParams at the first oscillation set.

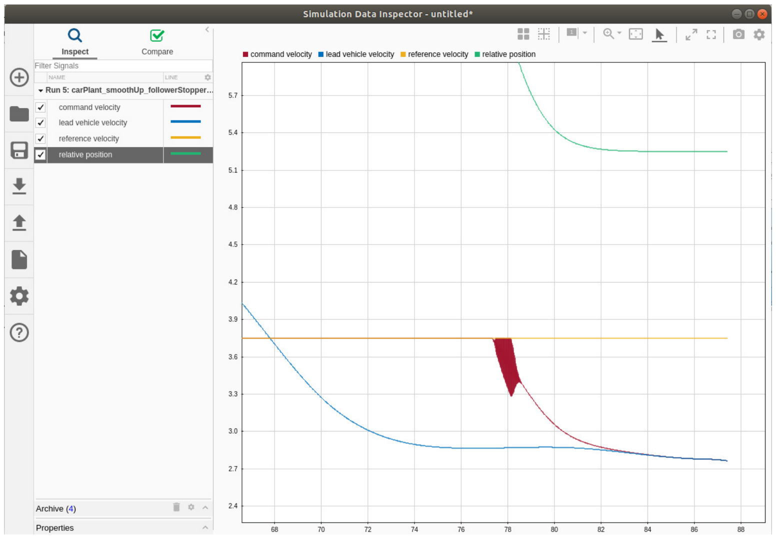


Figure 11: Original followerStopper and edited smoothUpParams at the second oscillation set.

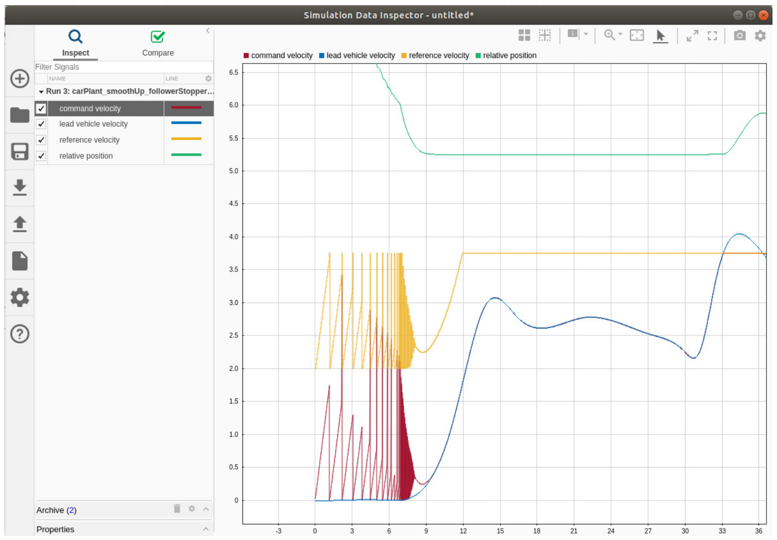


Figure 12: Edited (part 1) followerStopper and original smoothUpParams at the first oscillation set.

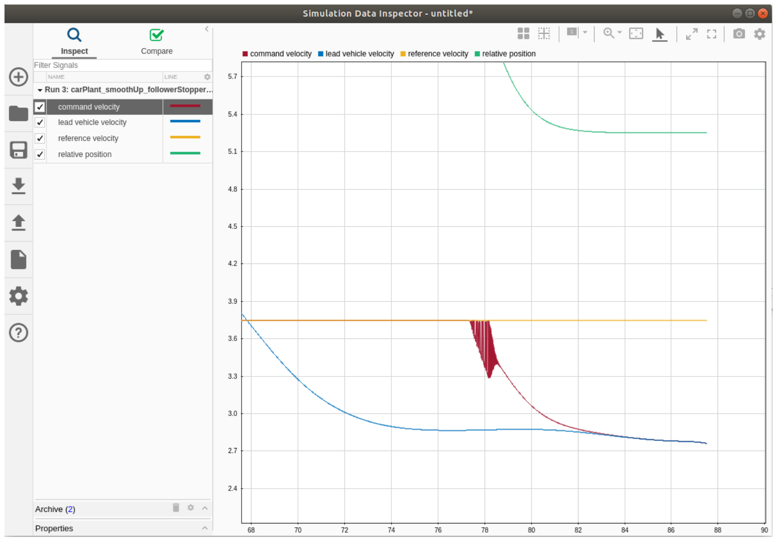


Figure 13: Edited (part 1) followerStopper and original smoothUpParams at the second oscillation set.

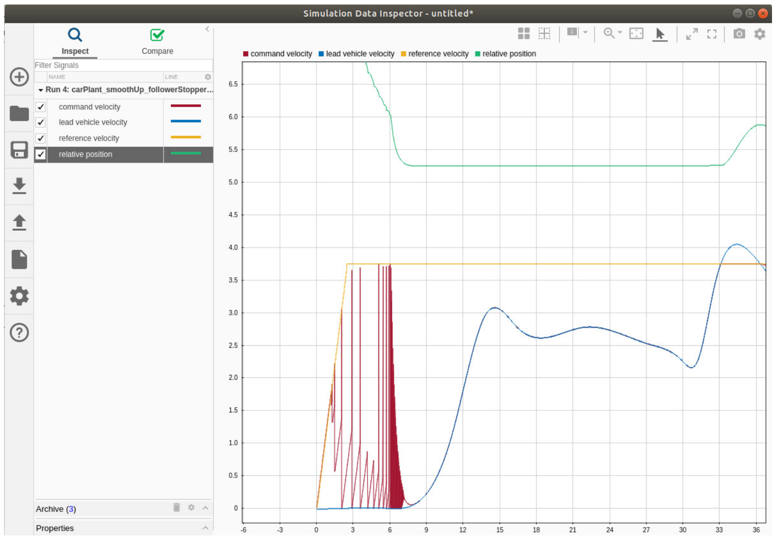

Simulation Data Inspector - untitled*
Inspect
Compare
Filter Signals
Run 4: carPlant_smoothUp_followerStopper...
command velocity
lead vehicle velocity
reference velocity
relative position
Archive (3)
Properties
■ command velocity ■ lead vehicle velocity ■ reference velocity ■ relative position

Figure 14: Edited (part 1) followerStopper and edited smoothUpParams at the first oscillation set.

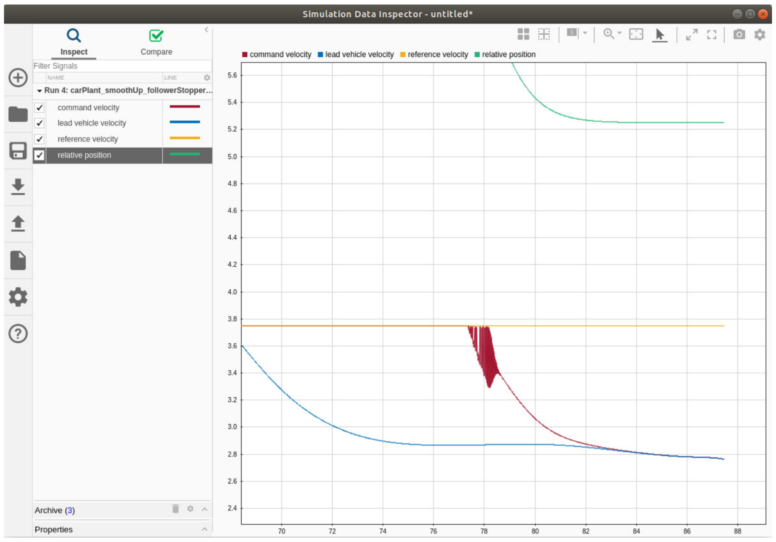
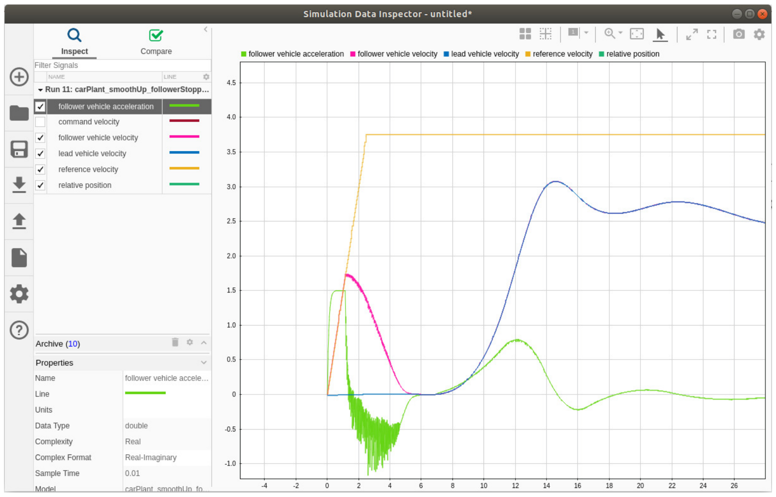
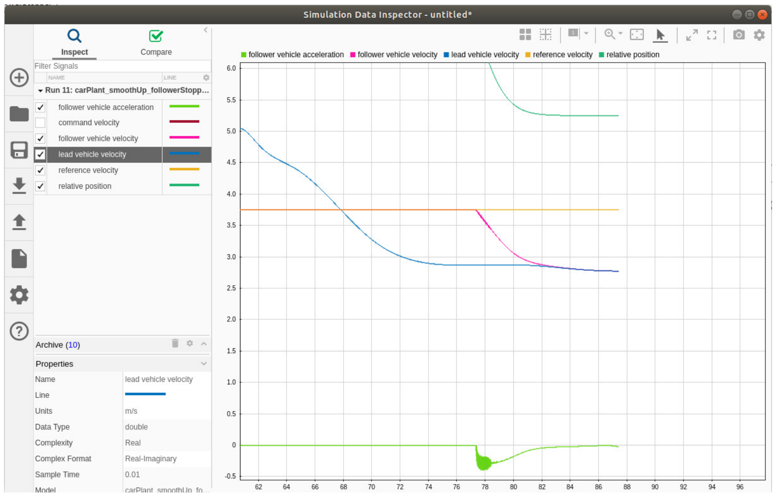

Simulation Data Inspector - untitled*
Inspect
Compare
Filter Signals
Run 4: carPlant_smoothUp_followerStopper...
command velocity
lead vehicle velocity
reference velocity
relative position
Archive (3)
Properties
■ command velocity ■ lead vehicle velocity ■ reference velocity ■ relative position

Figure 15: Edited (part 1) followerStopper and edited smoothUpParams at the second oscillation set.


Simulation Data Inspector - untitled*
Inspect
Compare
Filter Signals
Run 11: carPlant_smoothUp_followerStopp...
follower vehicle acceleration
command velocity
follower vehicle velocity
lead vehicle velocity
reference velocity
relative position
Archive (10)
Properties
🟩 follower vehicle acceleration 🟥 follower vehicle velocity 🟦 lead vehicle velocity 🟧 reference velocity 🟩 relative position

Name	follower vehicle accele...
Line	
Units	
Data Type	double
Complexity	Real
Complex Format	Real-Imaginary
Sample Time	0.01
Model	carPlant_smoothUp_fo...

Figure 16: Edited (part 2) followerStopper and edited smoothUpParams at the first oscillation set.


Simulation Data Inspector - untitled*
Inspect
Compare
Filter Signals
Run 11: carPlant_smoothUp_followerStopp...
follower vehicle acceleration
command velocity
follower vehicle velocity
lead vehicle velocity
reference velocity
relative position
Archive (10)
Properties
🟩 follower vehicle acceleration 🟥 follower vehicle velocity 🟦 lead vehicle velocity 🟧 reference velocity 🟩 relative position

Name	lead vehicle velocity
Line	
Units	m/s
Data Type	double
Complexity	Real
Complex Format	Real-Imaginary
Sample Time	0.01
Model	carPlant_smoothUp_fo...

Figure 17: Edited (part 2) followerStopper and edited smoothUpParams at the second oscillation set.