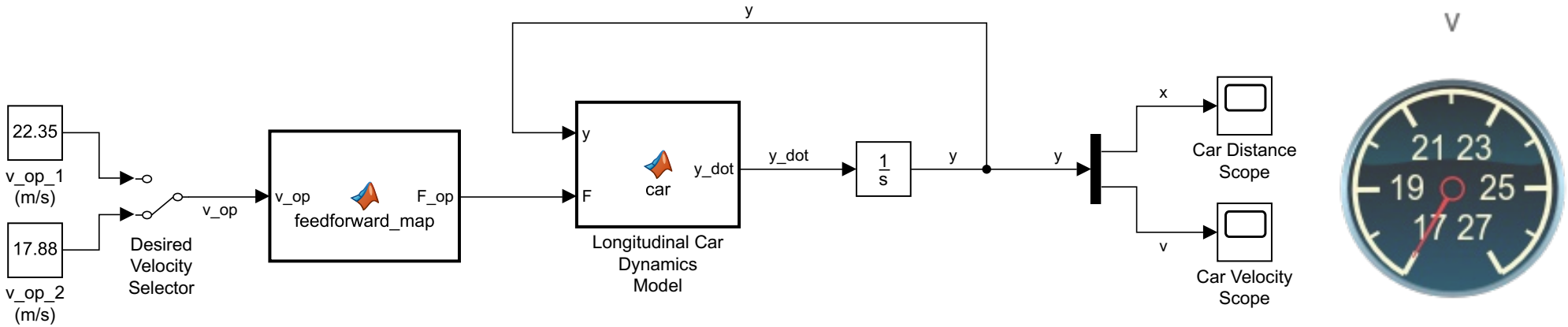
Open-loop System



1. To change the model parameters, select 'MODELING' from the pane up top, open the 'Model Settings' dropdown, and select 'Model Properties'. In the Model Properties dialog box, navigate to 'Callbacks' and select 'PostLoadFcn*', which is the function initializing the model parameters on when the model is done loading

* The simulation speed has been slowed down in order to allow the gauge to show the changing speed. Otherwise, the speed goes to the final value in an instant and the guage just jumps to the final speed
* There is no .m file to run before running this model

Open-loop Simulation

A nominal open-loop simulation has been run for two different nominal velocities with the model parameters set as follows:

$$m = 1000\ \text{Kg} \qquad b = 25\ {}^{\text{Ns}}/_{\text{m}} \qquad c_d = 0.1\ {}^{\text{kg}}/_{\text{m}}$$

These parameters can be changed by going to *Model Properties* in the task_5.xls file and navigating to the *PostLoadFcn* callback function

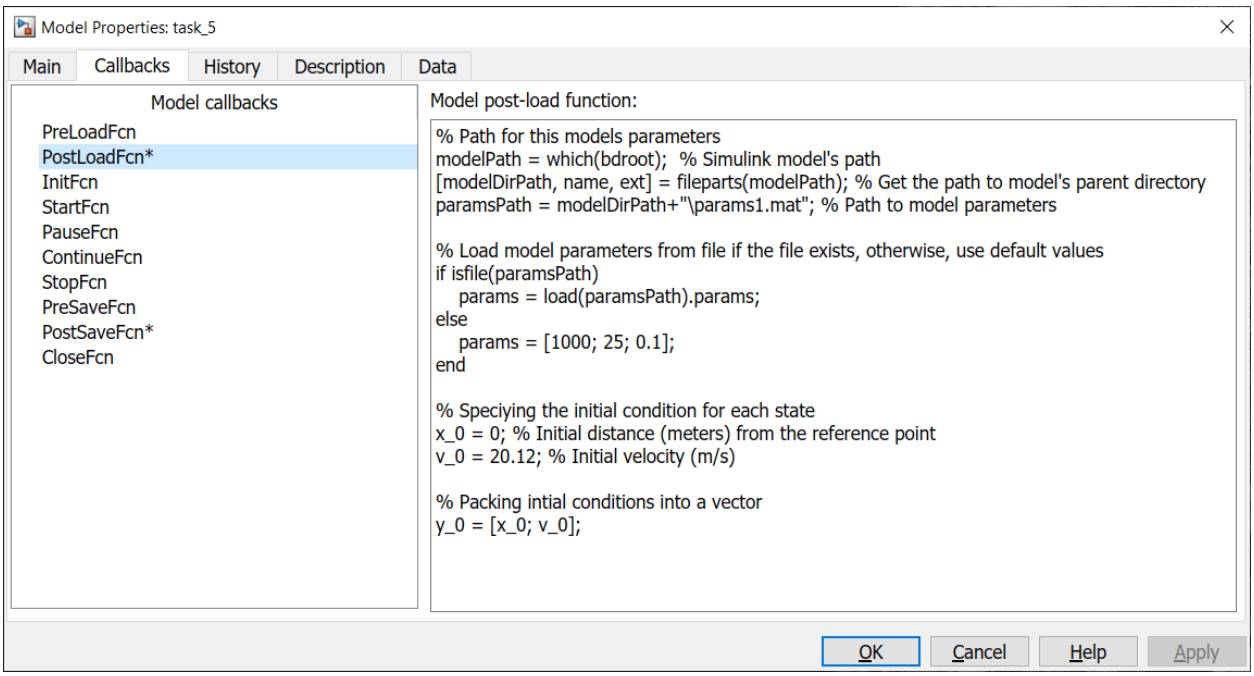


Figure 1: Model Configuration

Results for desired velocity set at $v_0 = 22.35\ {}^{m}/_{s}$:

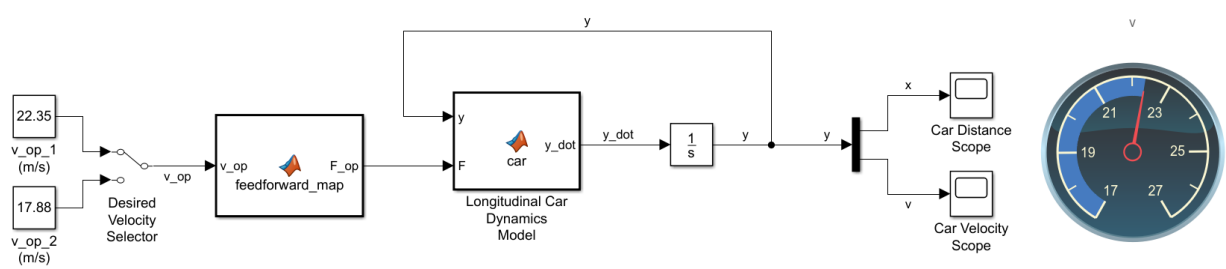
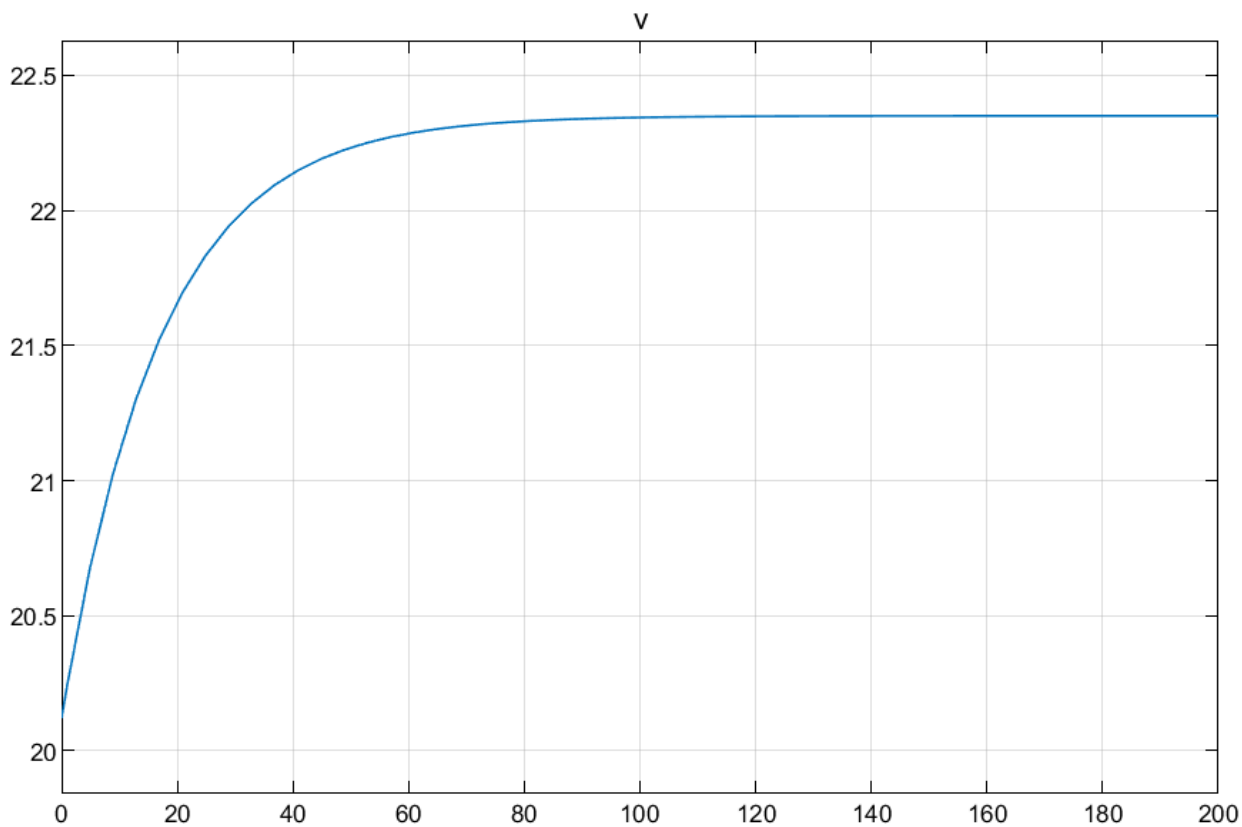


Figure 2: The Switch is flipped to allow the 22.35 m/s signal to go through

The velocity plot obtained is shown on the next page



It can be seen in the plot above that the car achieves the desired speed without being regulated by any controller and its response is stable, too. However, it takes too long a time to achieve that speed, which is where we need improvement.

Results for desired velocity set at $v_0 = 17.88\ {}^{m}/_{s}$:

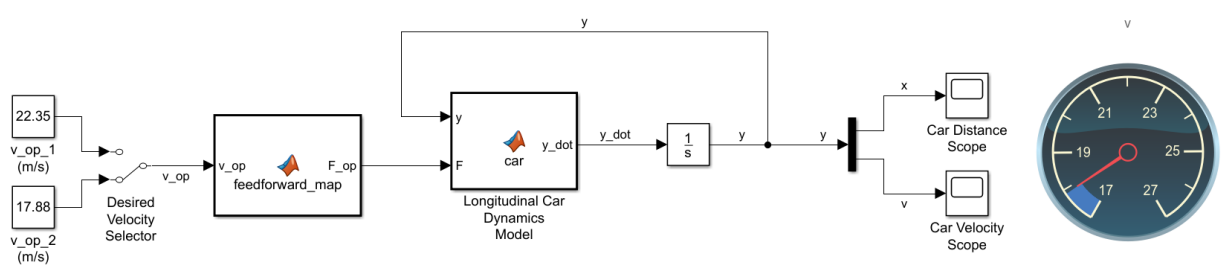
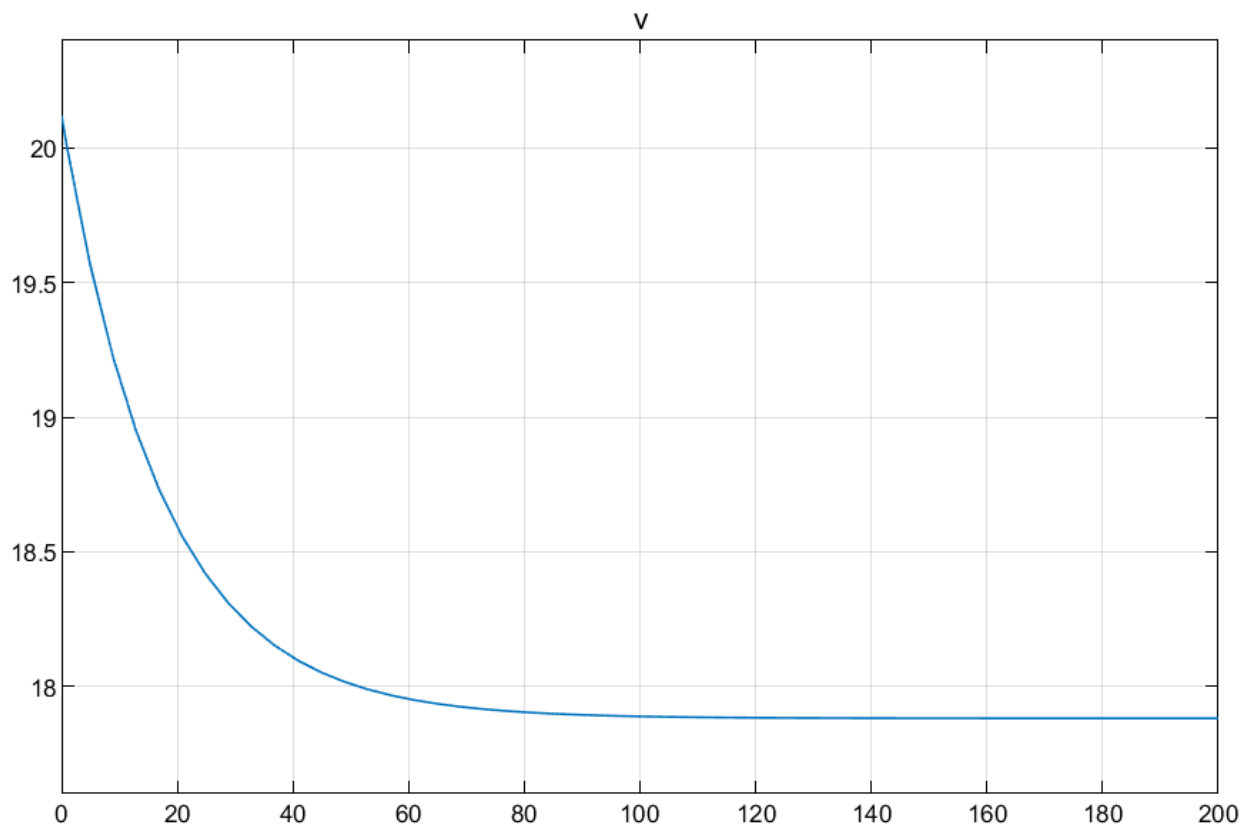


Figure 3: The Switch is flipped the other way to allow the 17.88 m/s signal to pass through

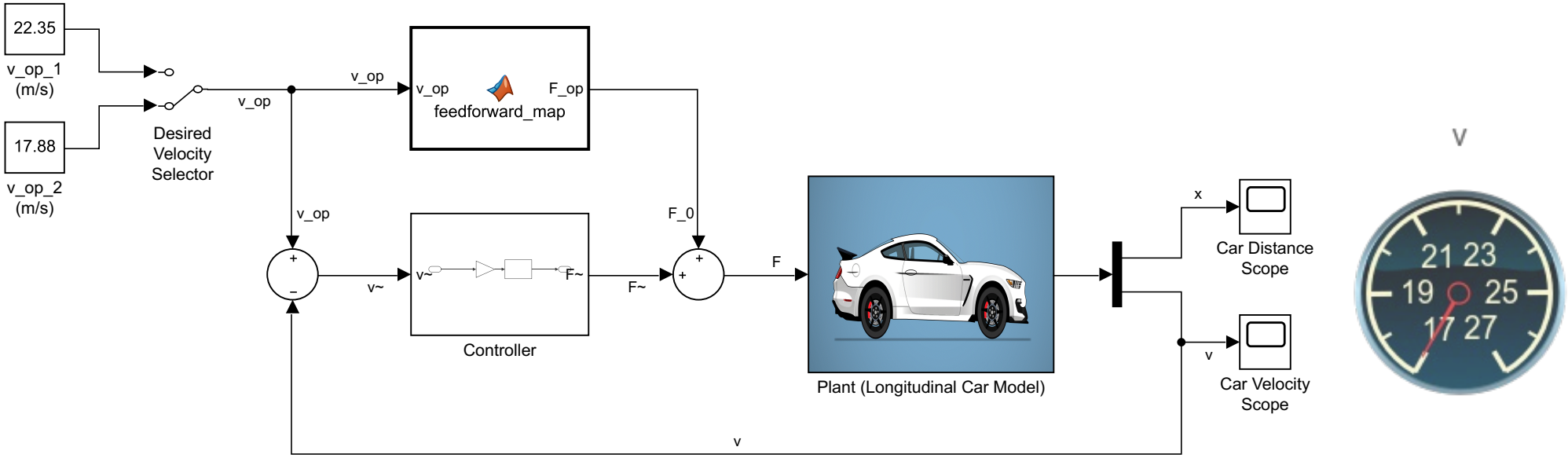
The velocity plot is given on the next page



This again shows that although the system is capable of achieving the desired velocity in open-loop configuration, it is too slow.

The SIMULINK file task_5.slx has a speedometer that shows the velocity as it changes in time, and it can be seen on it that the system takes quite a long time to get to the desired speed compared to the closed-loop setup presented next

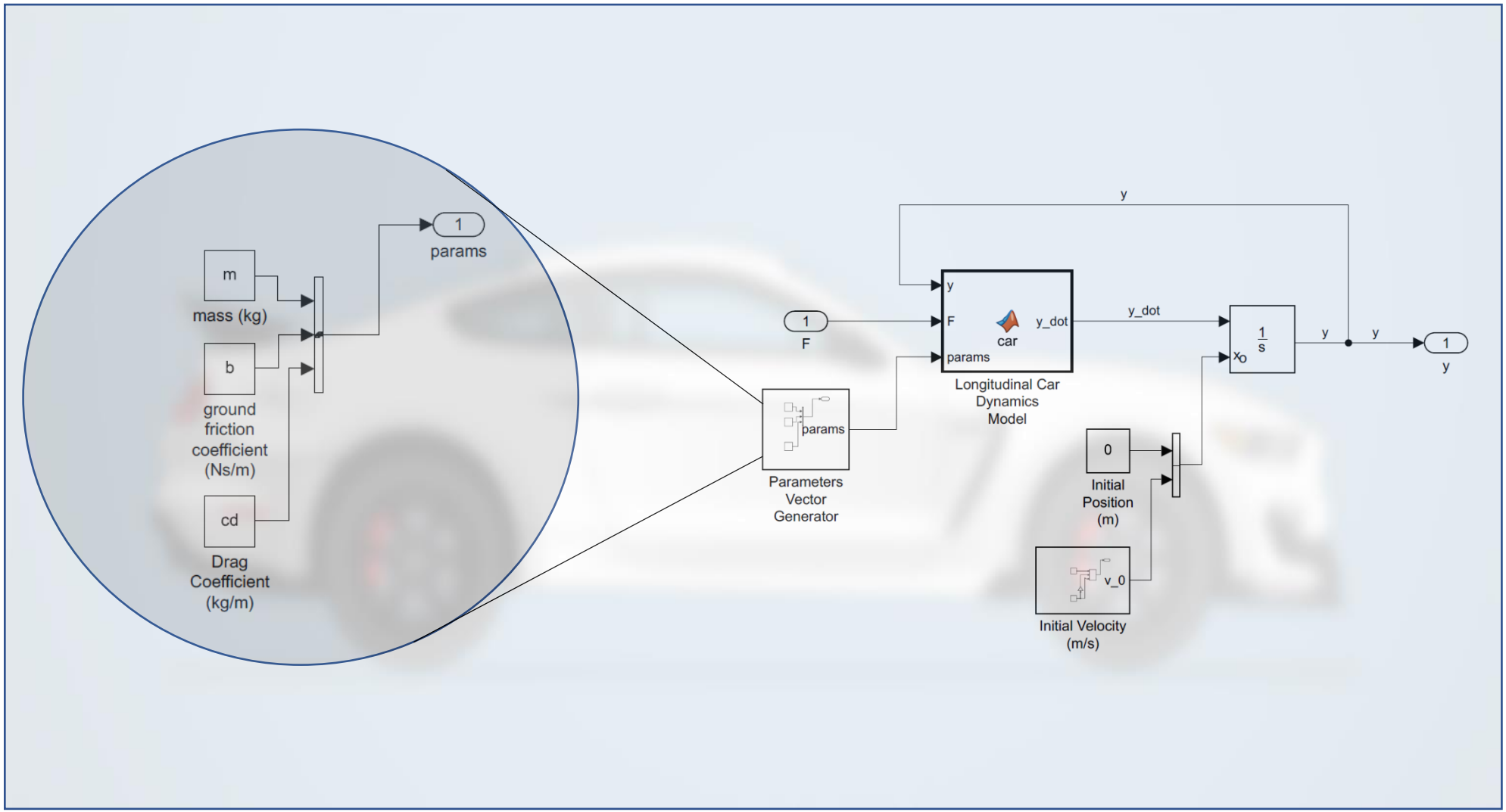
Closed-loop System



1. To change the model parameters, double click the Plant block (the one with the car) and enter the parameter values
2. To look inside the Plant block, select it and press Ctrl+Z

* The simulation speed has been slowed down in order to allow the gauge to show the changing speed. Otherwise, the speed goes to the final value in an instant and the guage just jumps to the final speed
* There is no .m file to run before running this model

Plant (Longitudinal Car Model)



Closed-loop Simulation

This time around, a closed-loop simulation has been run for the same nominal velocities as in the last section and with the same model parameters

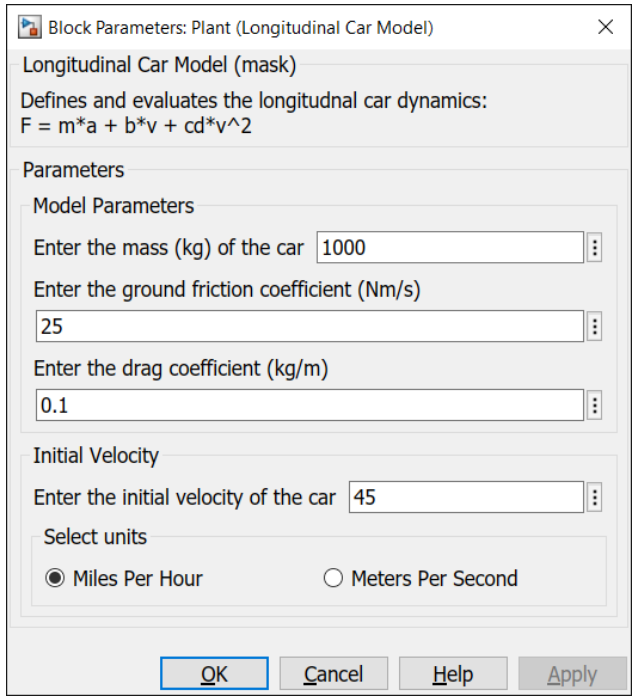


Figure 4: Model Configuration

Results for desired velocity set at $v_0 = 22.35\ {}^{m}/_{s}$:

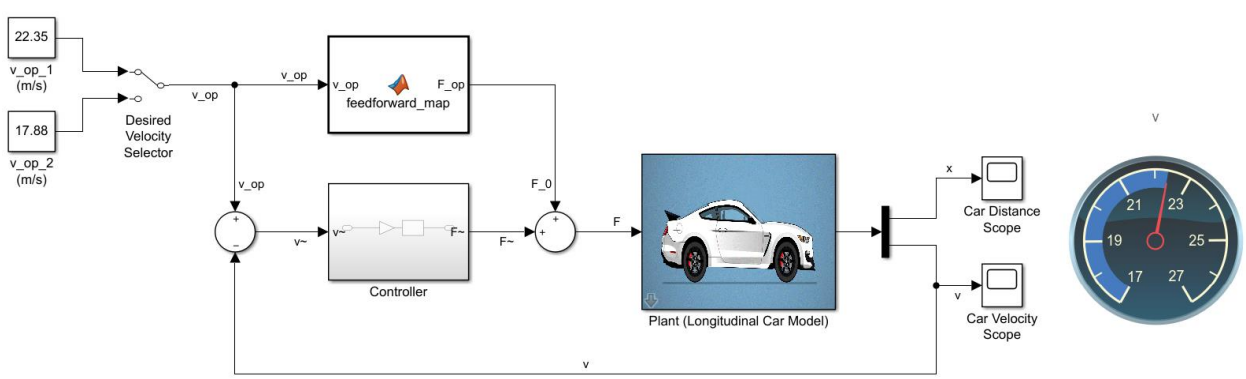
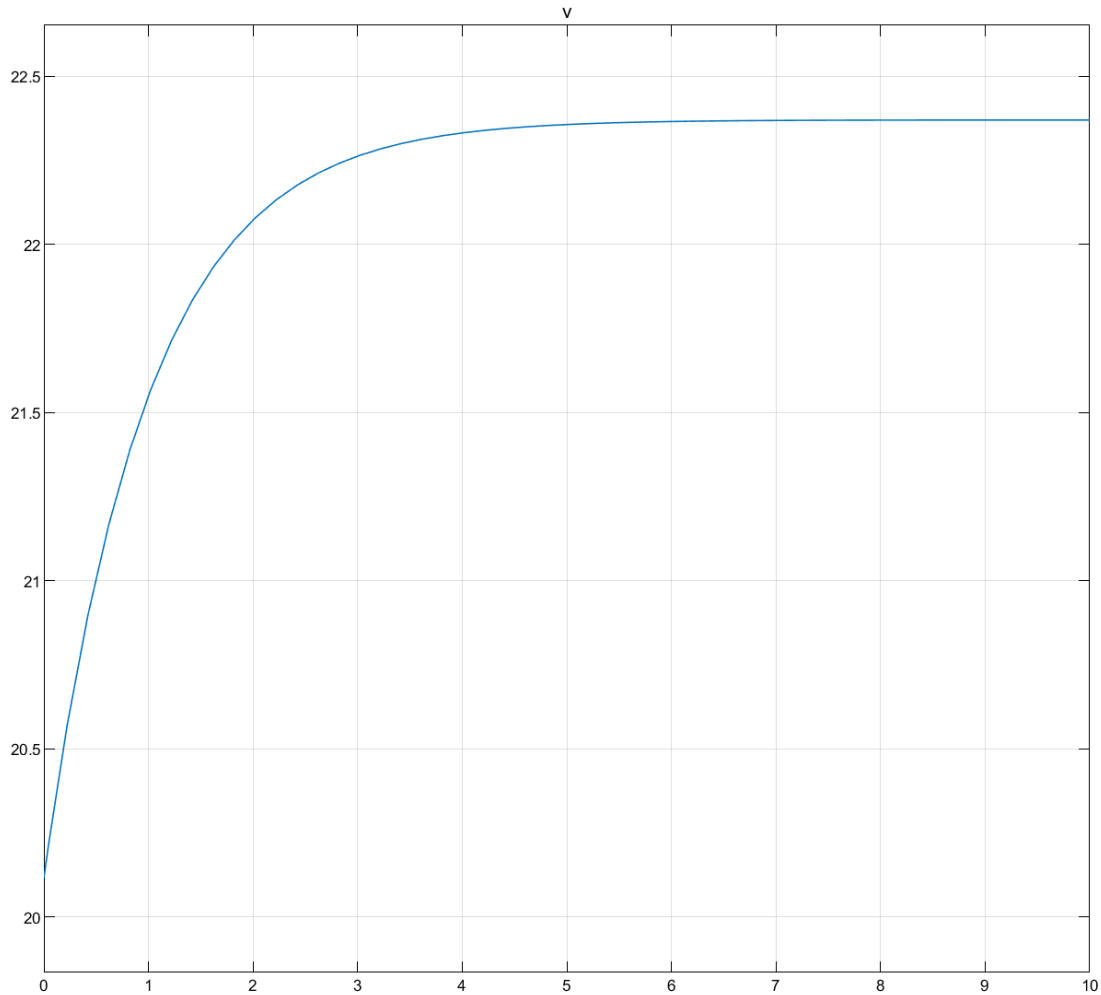


Figure 5: Closed loop system with the desired velocity set at 22.35m/s

With the plant in a feedback configuration now, the desired velocity is achieved much faster, as can be seen in the velocity profile below



Results for desired velocity set at $v_0 = 17.88\ ^{m}/_{s}$:

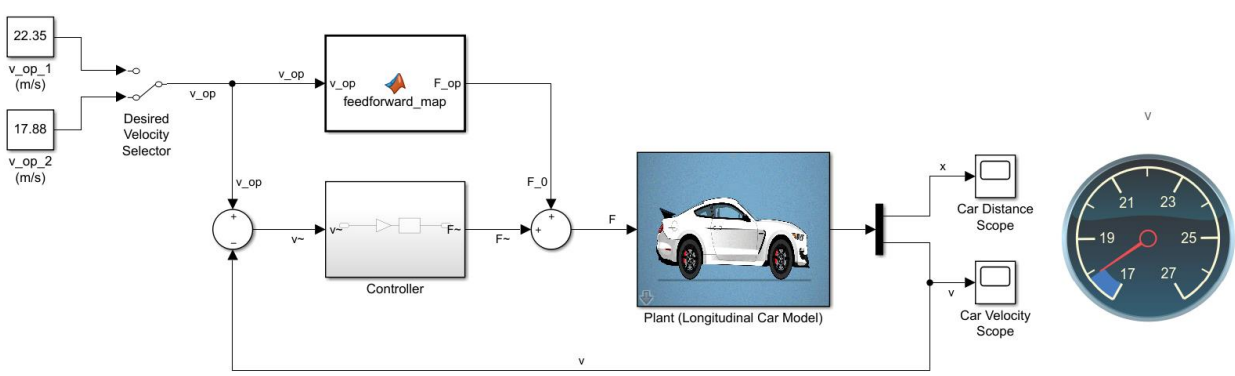
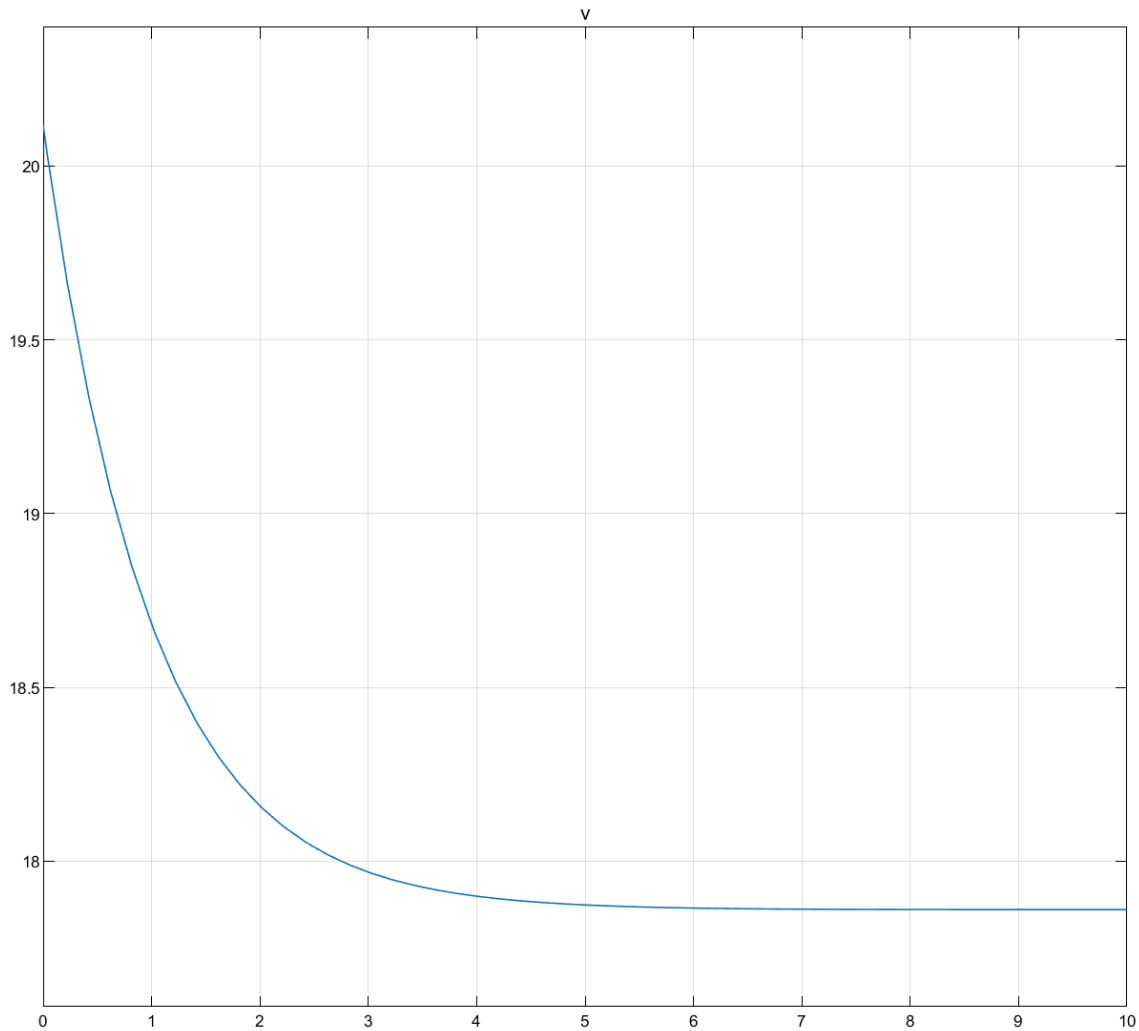


Figure 6: Closed loop system with the desired velocity set at 17.88m/s



This velocity plot tells the same story – that feedback has significantly improved the system's speed of response. This was the main design objective with the feedback control here and it has successfully been achieved with just PI control

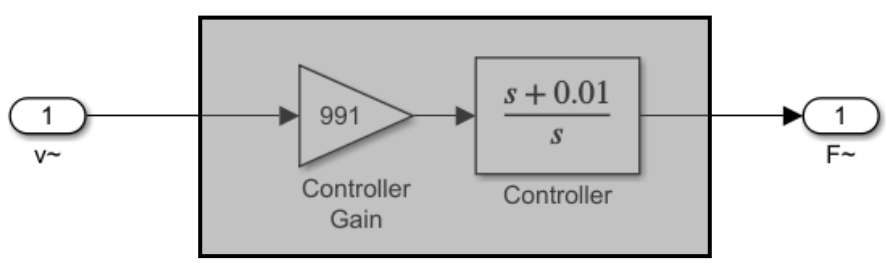
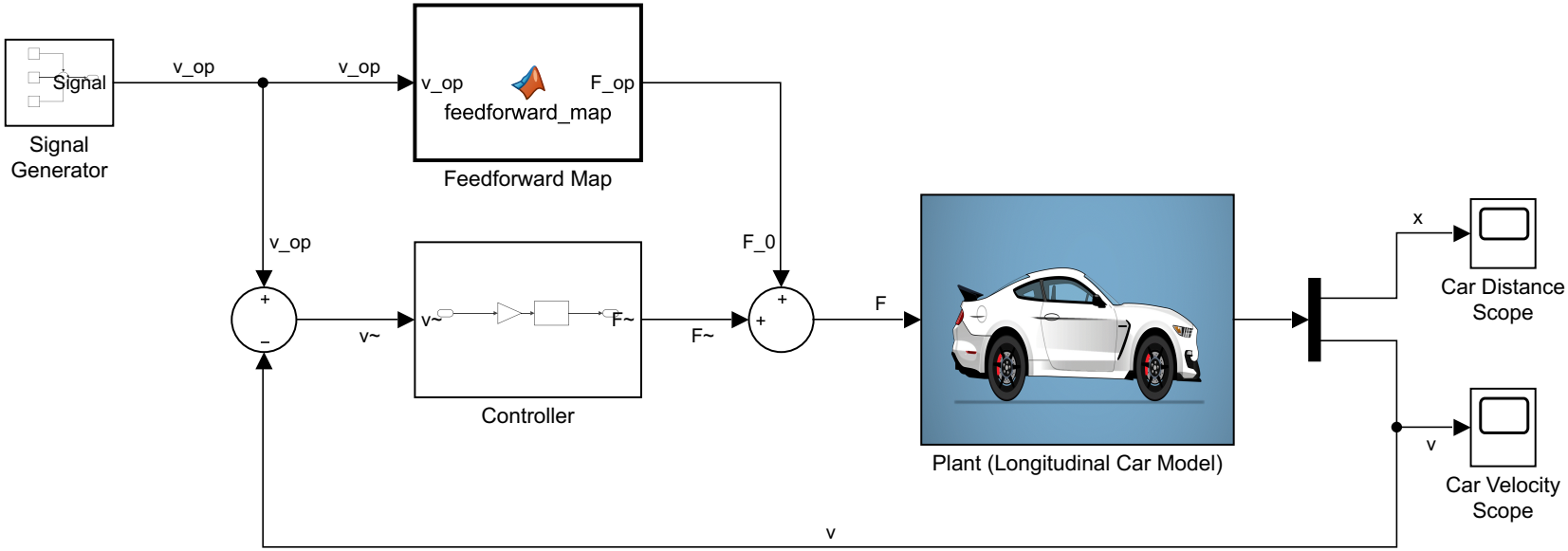


Figure 7: The PI controller and its input ($\tilde{v}$) and output ($\tilde{F}$)

Next, the closed-loop system is put to test by simulating it for a more realistic driving profile, where the desired speed is changing every 10 seconds

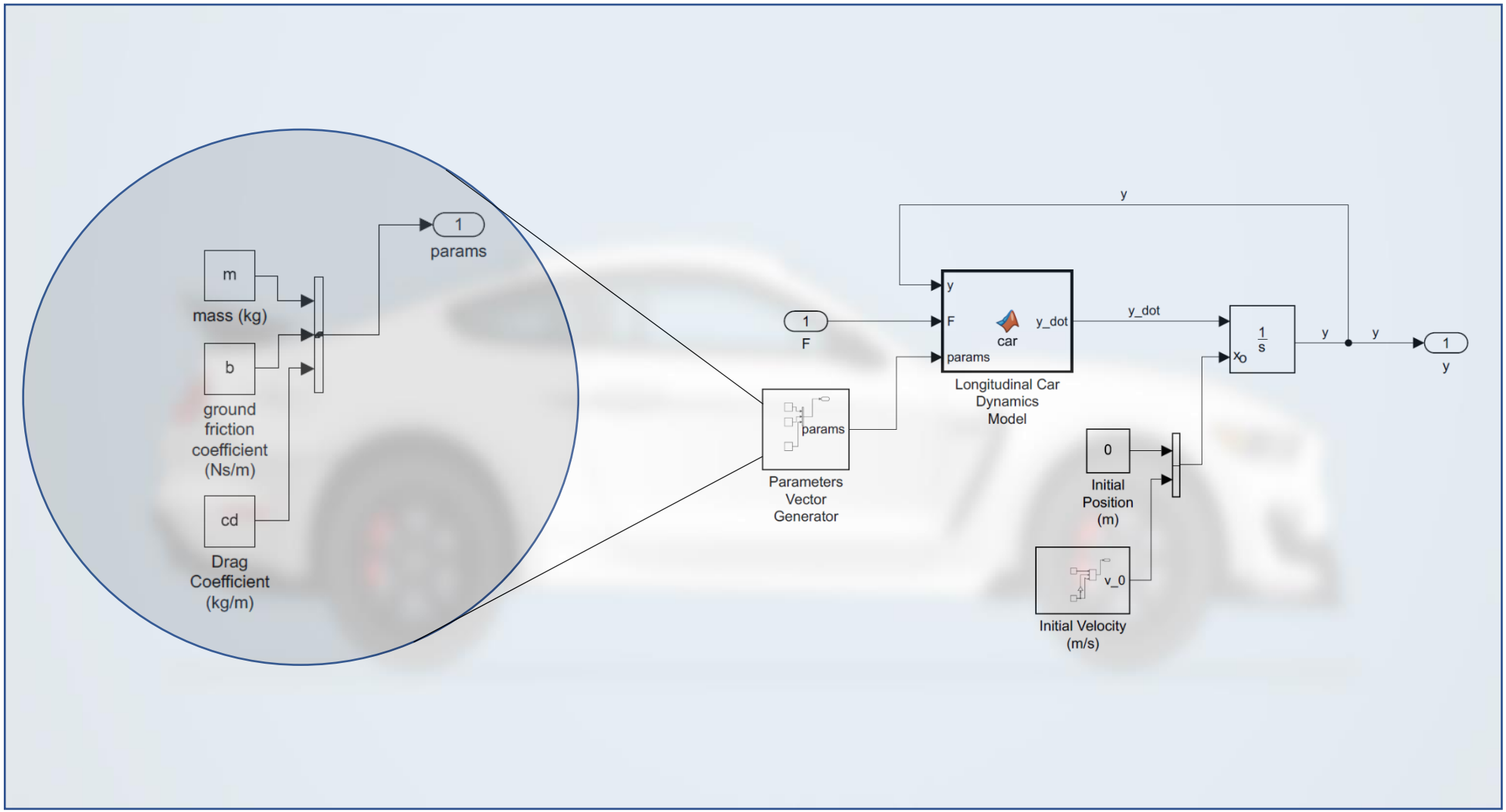
Closed-loop System with Driving Profile



1. To change the model parameters, double click the Plant block (the one with the car) and enter the parameter values
2. To look inside the Plant block, select it and press Ctrl+Z

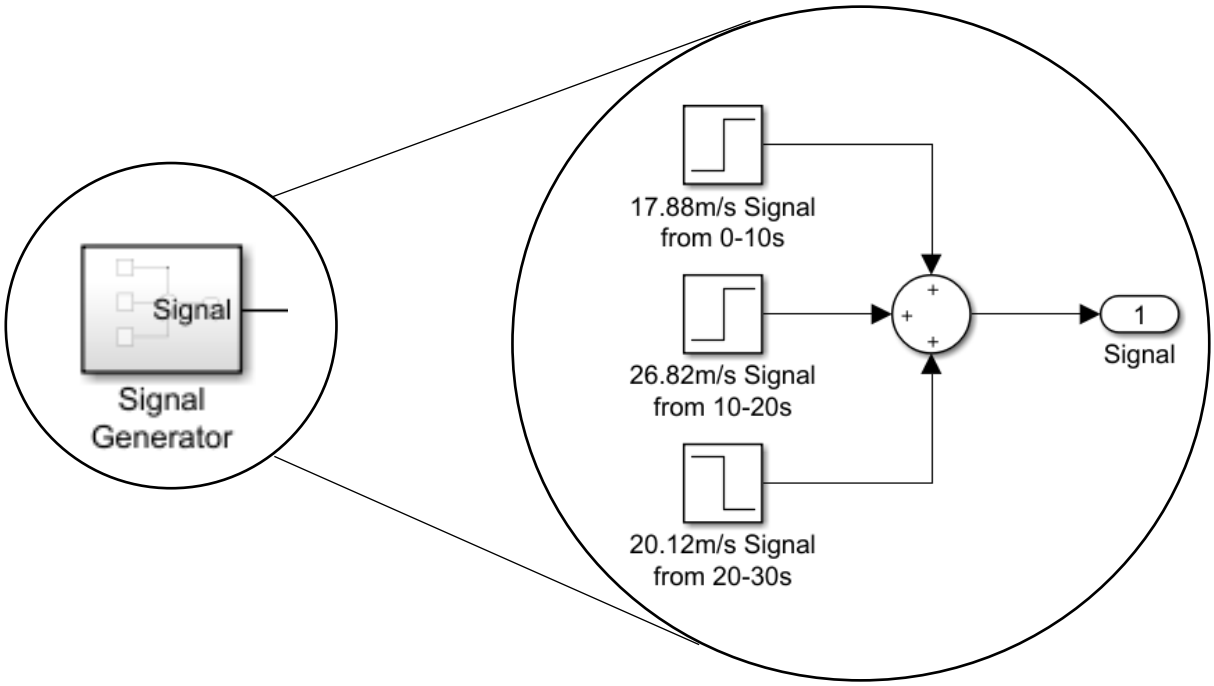
* The simulation speed has been slowed down in order to allow the gauge to show the changing speed. Otherwise, the speed goes to the final value in an instant and the guage just jumps to the final speed
* There is no .m file to run before running this model

Plant (Longitudinal Car Model)



A somewhat realistic test

A more realistic-looking driving driving profile has been produced through a custom signal generator implemented as follows:



It has been used below to test both the open-loop and the closed-loop system configurations

Results for open-loop system

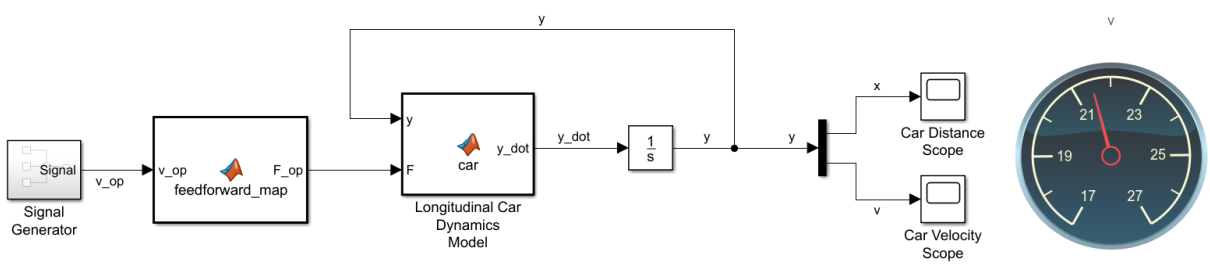
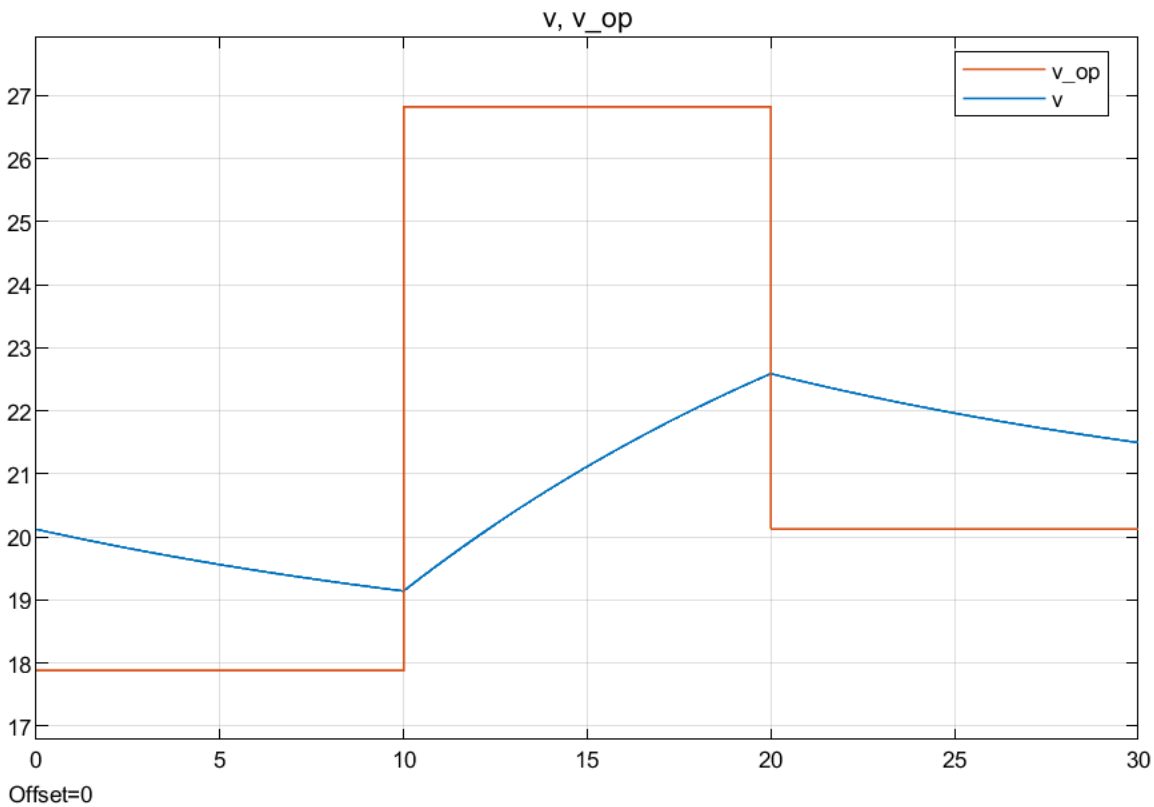


Figure 8: Closed loop system subjected to our "realistic" driving peofile

The velocity plot (shown on the next page) clearly shows how the slow response speed of the open-loop system is keeping it from tracking the desired velocity profile. Before the system is able to hit the intended speed, the speed changes, and therefore, it is not able to track it



Results for closed-loop system

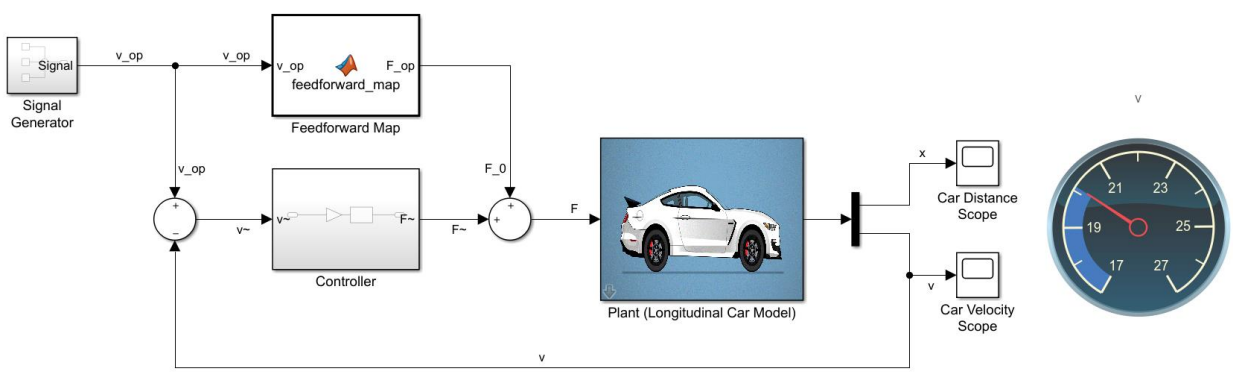
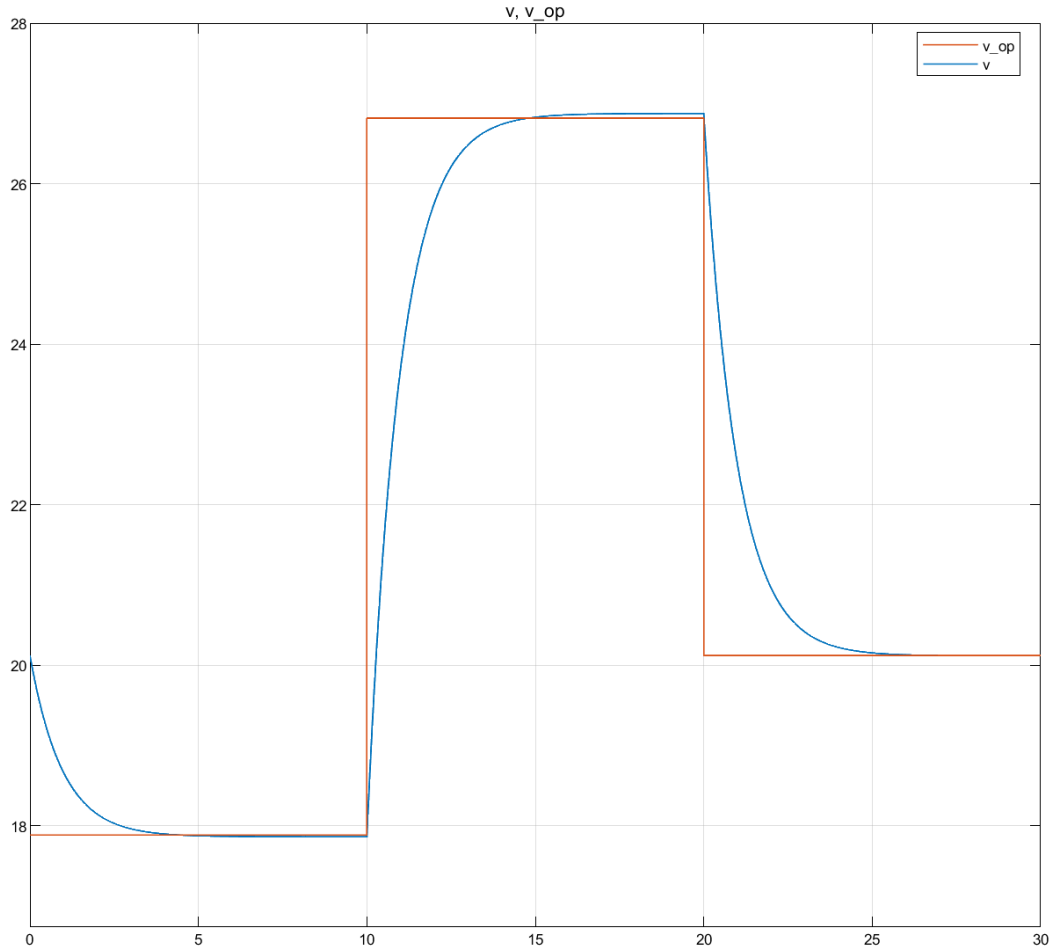


Figure 9: Closed loop system subjected to our "realistic" driving profile

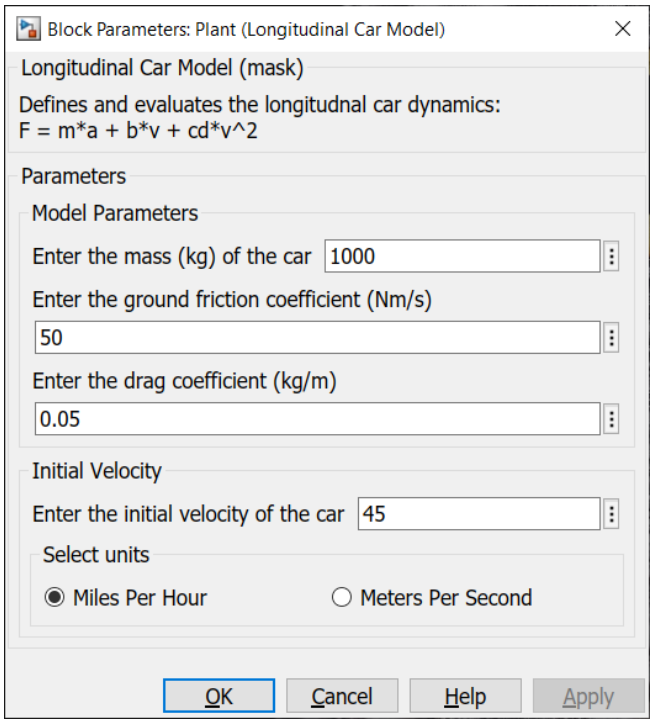
The closed loop system is tested on the same driving profile and it is able to keep track of it much better. The velocity plot on the next page outlines this clearly

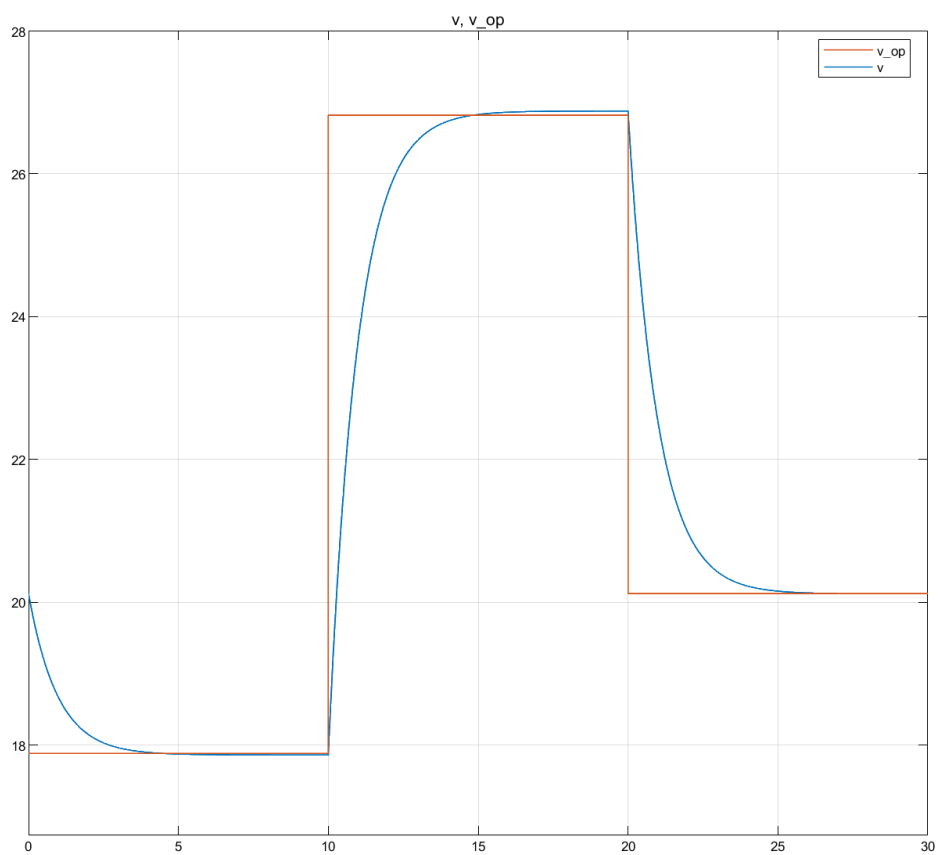


The plot shows there is a slight steady state error in the tracking, which is particularly visible when the 26.82 m/s signal is tracked. This is probably due to the fact that the controller is based on a linearized model of the plant, and as we start to track speeds in the region where the linearized model does not hold as accurately, the steady-state error starts to appear

These results were achieved for a nominal plant model, but in reality, the air drag and other factors might not be nominal. So, it is worthwhile to investigate if our system is robust to changes in the plant itself, which will be taken up next.

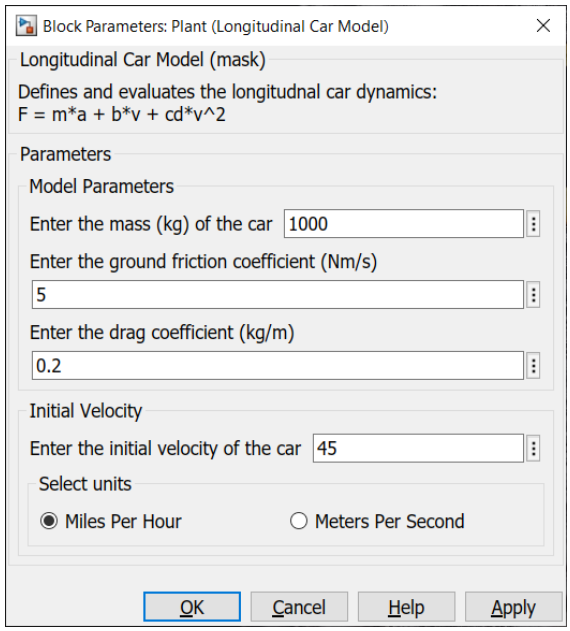
Let us change the plant parameters to reflect conditions with high ground friction and low air drag and rerun the simulation

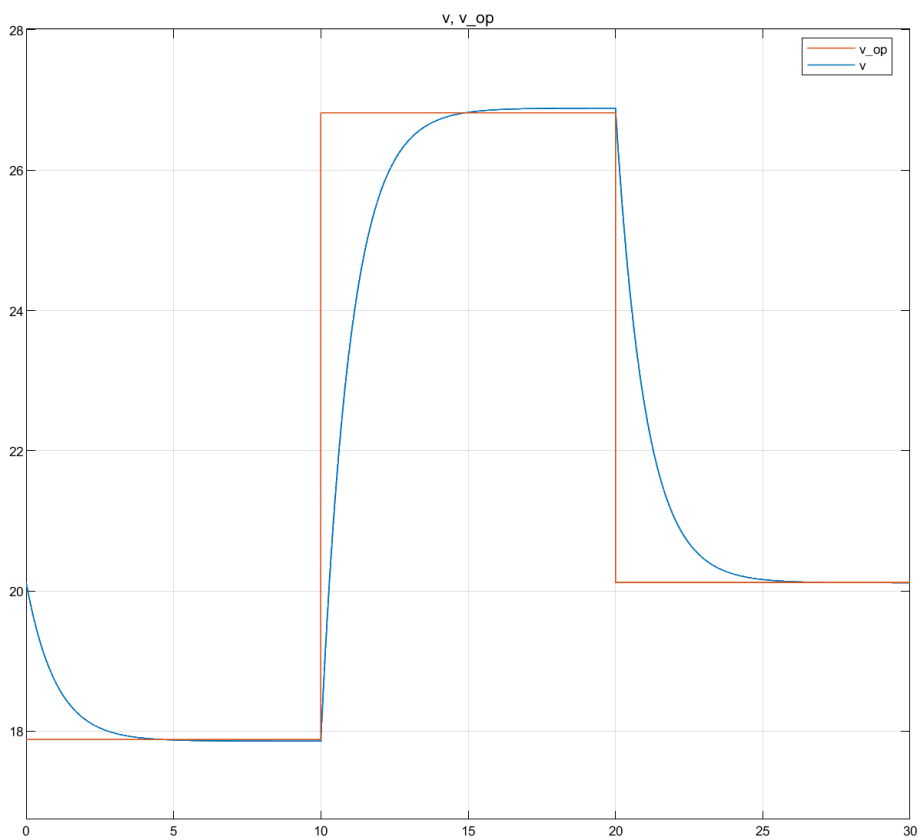




The response is, once again, very desirable.

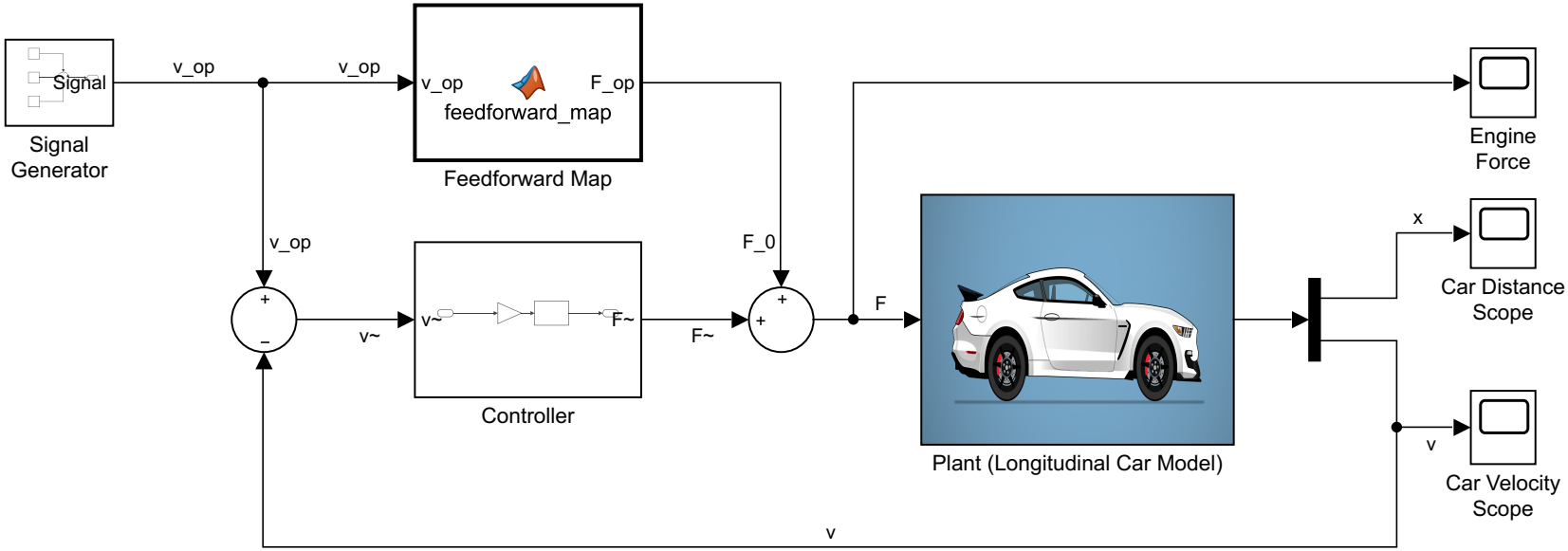
Let's check for conditions when the ground friction is low and the air drag is high





The system has again tracked the velocity profile very accurately

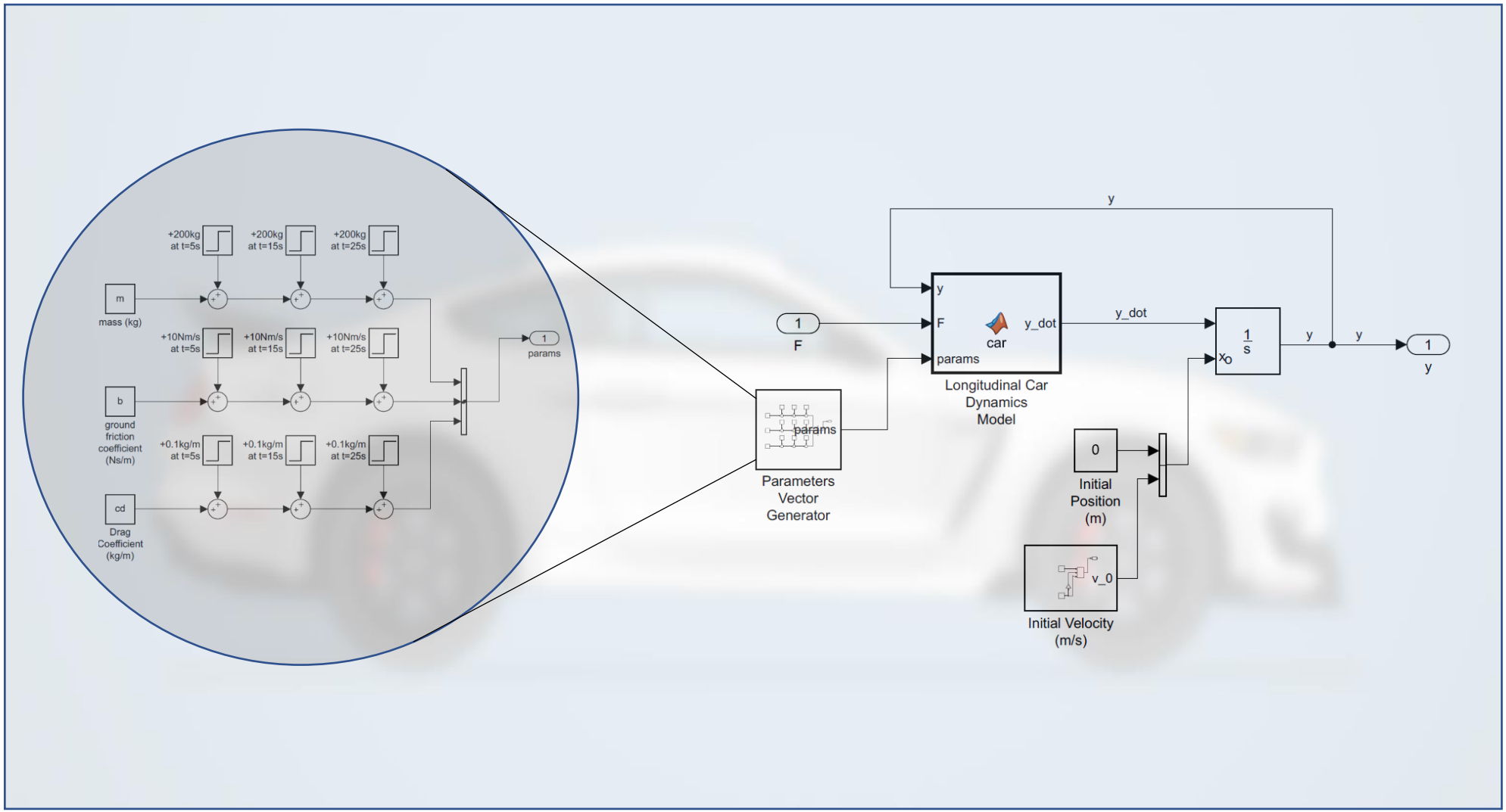
Affect of Changes in Model Parameters on the Closed-loop System



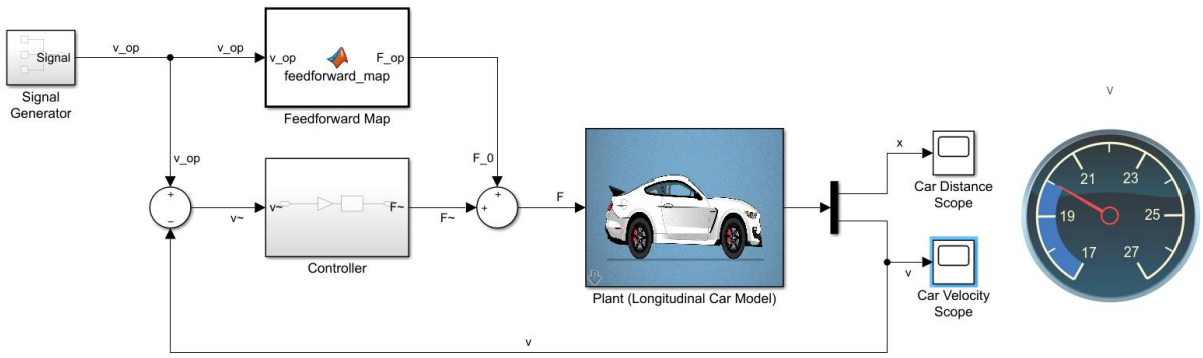
1. To change the model parameters, double click the Plant block (the one with the car) and enter the parameter values
2. To look inside the Plant block, select it and press Ctrl+Z

* The simulation speed has been slowed down in order to allow the gauge to show the changing speed. Otherwise, the speed goes to the final value in an instant and the guage just jumps to the final speed
* There is no .m file to run before running this model

Plant (Longitudinal Car Model)

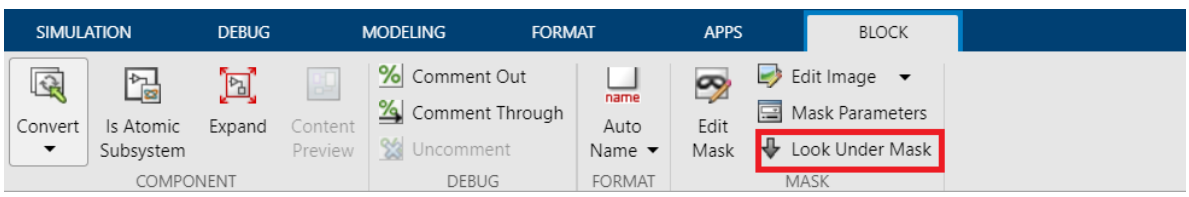


What if the model parameters are changing over time? Is our controller robust to that too? For this task, the system configuration is similar to the last one:

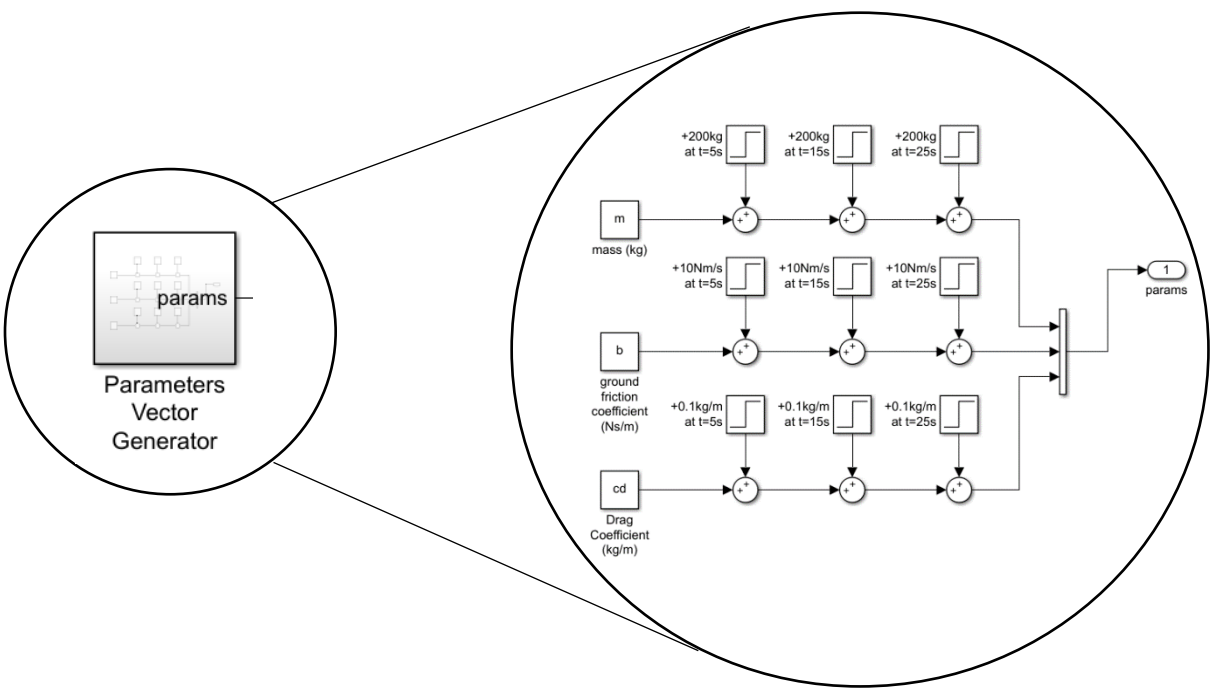


But there is one difference. The Plant is evolving in time!

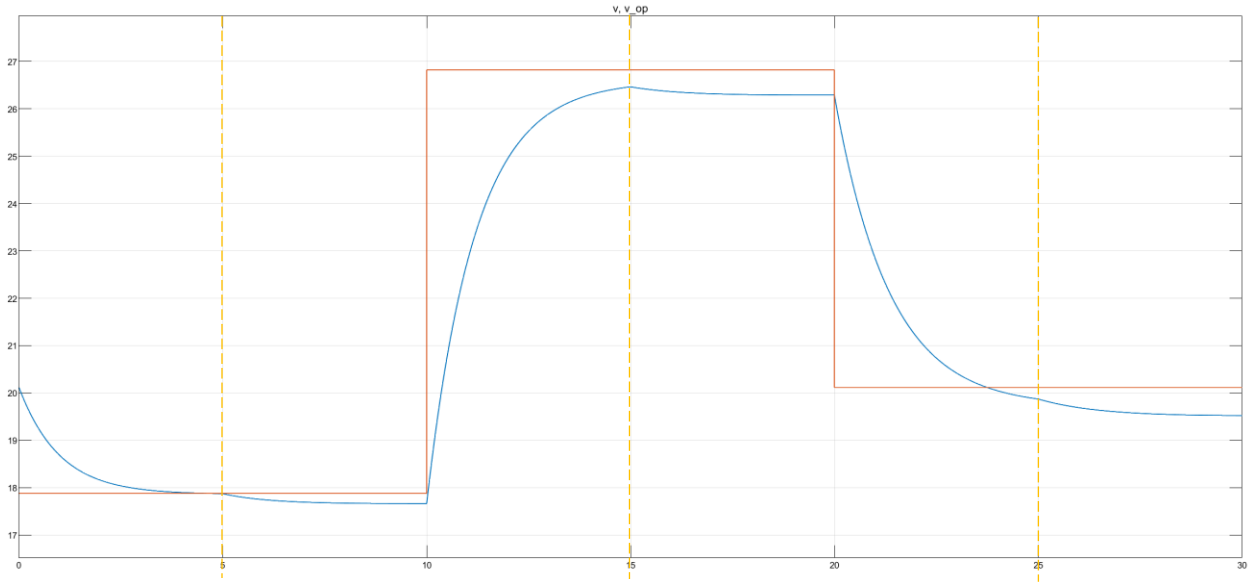
To look inside the plant block, please select it, and from the pane at the top, select "Look under Mask"



You will be able to see the parameter-generator block, which changes the model's parameters over time



When the simulation is run, it turns out the controller is unable to track the velocity reliably



Plant model changed

$m = 1200\ Kg$
$b = 35\ {}^{Ns}/_{m}$
$c_d = 0.2\ {}^{kg}/_{m}$

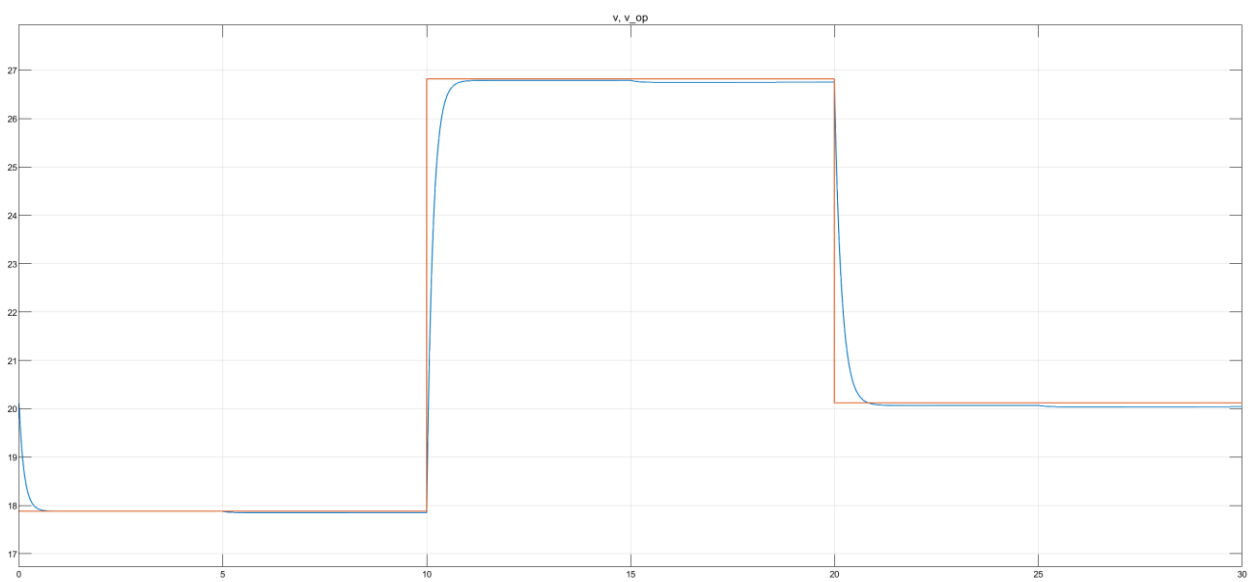
Plant model changed

$m = 1400\ Kg$
$b = 45\ {}^{Ns}/_{m}$
$c_d = 0.3\ {}^{kg}/_{m}$

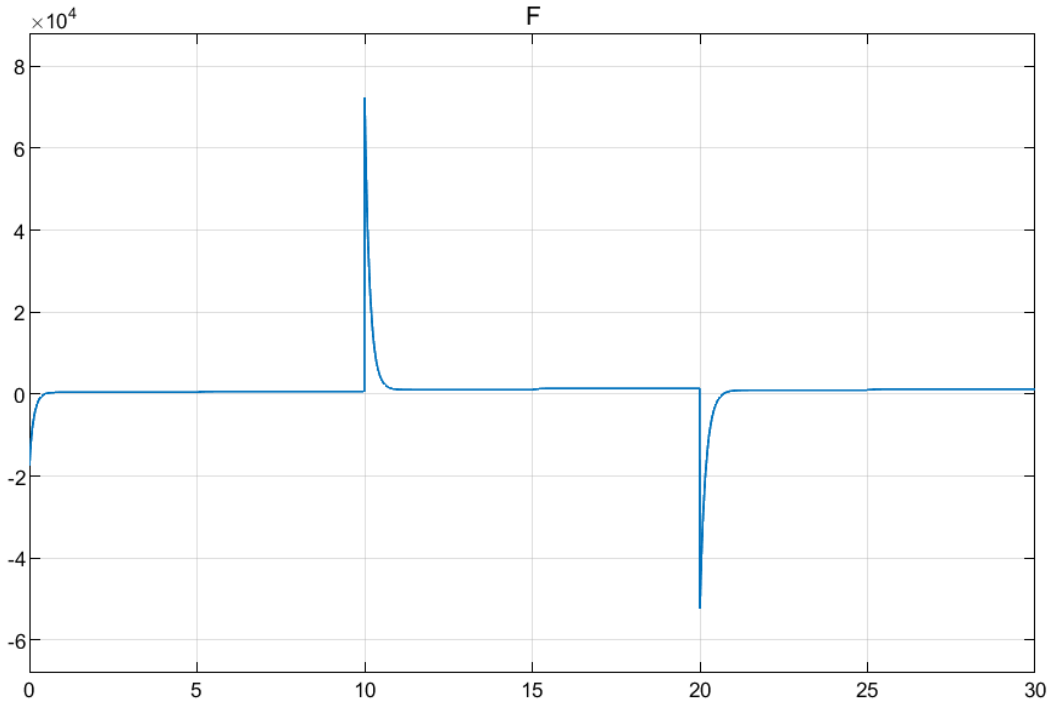
Plant model changed

$m = 1600\ Kg$
$b = 55\ {}^{Ns}/_{m}$
$c_d = 0.4\ {}^{kg}/_{m}$

It seems like just when the controller is about to settle into the velocity profile, either the plant changes or the velocity profile changes. One way to fix this would be to have a high gain for the controller. If the controller gain is increased from 991 to 8000, a significant improvement can be observed (as shown below)



Increasing the gain too much can, however, lead to engine torques that might very well be too wishful. Indeed, if we have a look at the force profile for the engine, we get two sharp spikes exceeding 70,000 Newtons. This figure is up there with the thrust produced by jet engines.[i]



It would be interesting to try other control architectures, especially PID, and see how well they perform with the plant dynamics changing, as well as consider the affect of uncertainty and sensor noise. But because of time constraints, this has not been taken up here.

[i] https://www.ge.com/reports/the-worlds-largest-jet-engine-is-already-more-powerful-than-americas-first-manned-space-rocket/

APPENDIX

Feedforward Map

(MATLAB Function Block)

v_{op} →

```
function F_op = feedforward_map(v_op, params)
    %% Specifies mapping from nominal velocity to nominal force
    % Inputs:
    % -------
    % v_op: The nominal velocity (velocity at equilibrium point)
    % params: A vector of system parameters
    %         It is a 3-dimensional column vector of the form:
    %         [mass; ground_friction-coefficient; drag_coefficient]
    %
    % Output:
    % -------
    % F_op: The nominal force (force at equilibrium point)

    % Unpacking Parameters
    m = params(1)
    b = params(2)
    cd = params(3)

    % Defining feedforward mapping from v_op to F_op
    F_op = b*v_op + cd*v_op^2;

end % end feedforward_map function
```

→ F_{op}

Feedforward Map

Longitudinal Car Dynamics Model

(MATLAB Function Block)

y

F

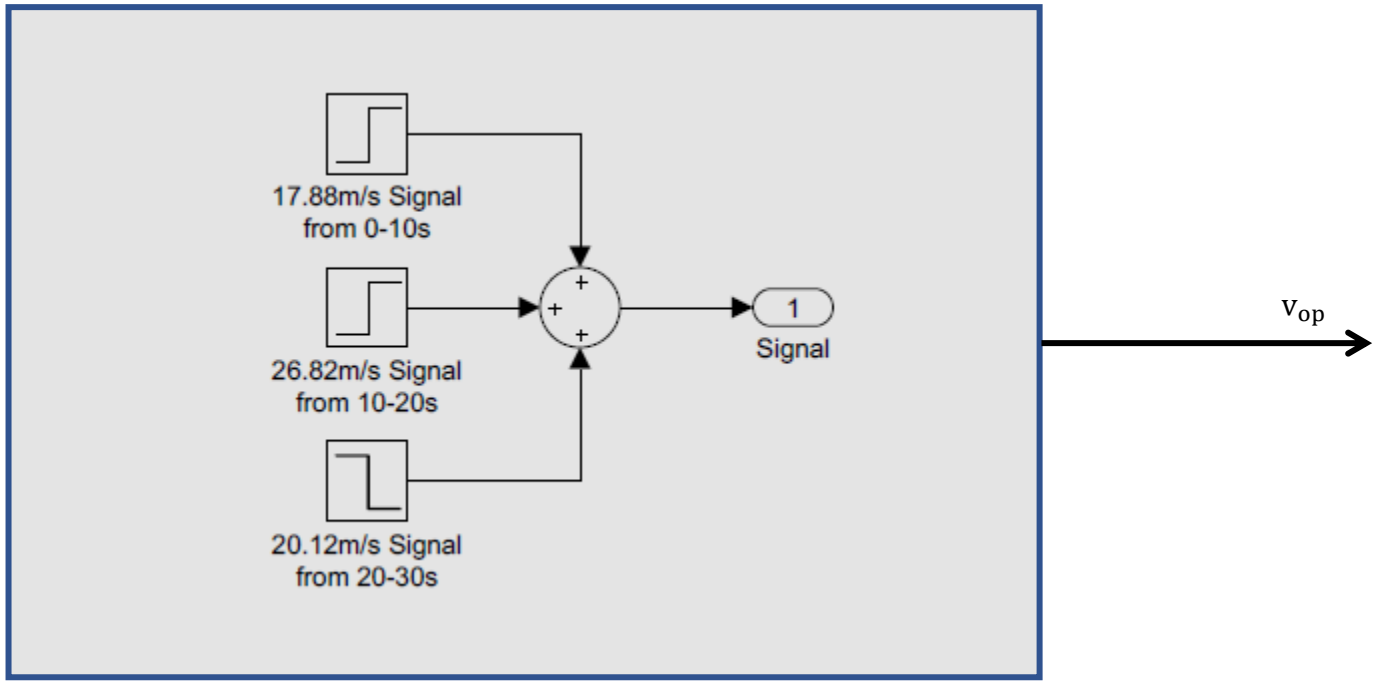
$params$

```
function y_dot = car(y, F, params)
    %% This subroutine defines the longitudal car dynamics
    % Inputs
    % ------
    % y: State vector
    %    It is a 2-dimensional column vector of the form:
    %    [distance; car_velocity]
    % F: Input force (acting on the car)
    % params: A vector of model parameters
    %         It is a 3-dimensional column vector of the form:
    %         [mass; ground_friction-coefficient; drag_coefficient]
    %
    % Outputs
    % -------
    % y_dot: Time derivative of the state vector
    %        It has the same dimension as the state vector (y) and is
of
    %        the form:
    %        [car_velocity; acceleration];

    % Unpacking parameters
    m = params(1);
    b = params(2);
    cd = params(3);

    % Defining the ODEs describing system's dynamics
    y_dot_1 = y(2); % ODE defining car velocity
    y_dot_2 = (F-b*y(2)-cd*y(2)^2)/m; % ODE defining car acceleration

    % Pack the differentiated states into a vector
    y_dot = [y_dot_1; y_dot_2];

end % end car subroutine
```

$\dot{y}$

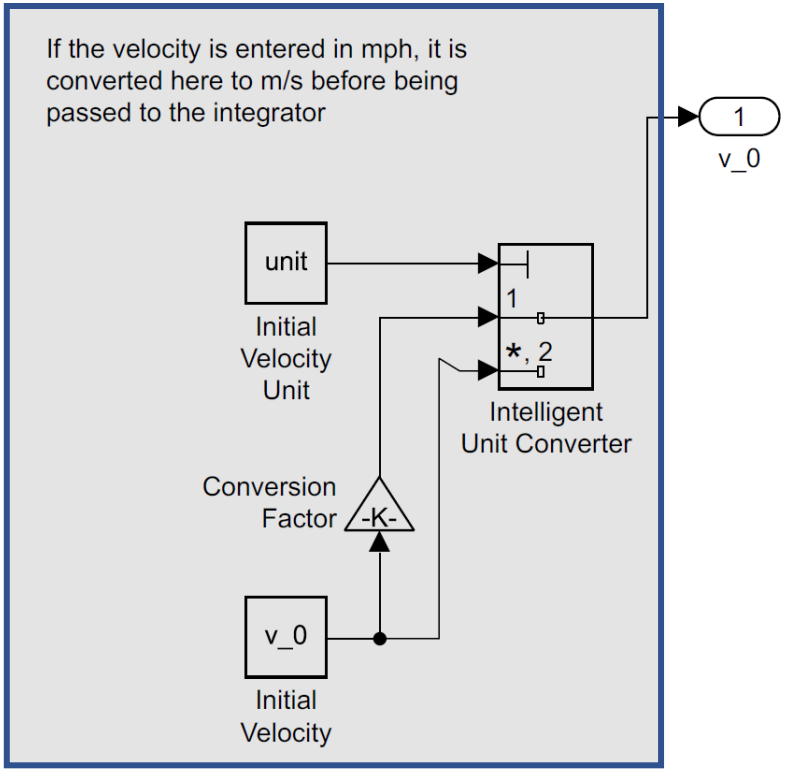
Longitudinal Car Dynamics Model

Signal Generator



Signal Generator

Initial Velocity Block



Initial Velocity