

# Driver Controls

Configurable Functions .....	2
Documentation for Configurable Functions .....	2
Scaling reference waveforms .....	4
Working with Multiple Datasets for Configurable Functions .....	5
Open-Loop Controls .....	6
Replacing Open-Loop Variables with Custom Controller Outputs .....	6
Summary of Open-Loop Control Functions and Variables .....	7
Reference Paths .....	8
Defining Reference Paths .....	9
Paths and LTARG Datasets Assigned to the Vehicle .....	10
Changing Paths Using Events .....	11
The Built-in Closed-loop Driver Model (DM) .....	13
Steering by Torque .....	14
DM Coordinate System .....	15
Steer Control Using Geometry and a Single Preview Point .....	17
Steer Control using Linear Dynamics and Multiple Preview Points .....	20
Viewing the DM Preview Point(s) .....	23
Using DM with External Models .....	24
The Auxiliary Steering Controller .....	24
Trailer Backing Controller .....	25
Steering by the Closed-Loop Driver Model: GUI Screen .....	25
Lateral Target .....	25
DM Settings .....	26
Visualization of the Target Path in VS Visualizer .....	30
Working with Multiple LTARG and Path Datasets .....	31
The Built-in Closed-Loop Speed Controller (SC) .....	31
Installing the Speed Controller .....	32
Interactions with Open-Loop Brake and Throttle Commands .....	34
Operation of SC .....	34
Setting Speed from Run Control, Procedures, and Events Screens .....	42
Speed (Closed Loop) Using Target Speed: GUI Screen .....	44
Speed (Closed Loop) Using Path Preview: GUI Screen .....	47
Vehicle Tests with Many Changes in Speed Target .....	56
Speed Control for Starting, Stopping, and Reversing .....	57
Shift Control .....	62
Specifying Closed-Loop Control of Transmission Gear .....	63
Specifying Transmission Gear with an Open-Loop Function .....	64
Closed-Loop Shifting by Clutch .....	65
Other Applications of the Closed-Loop Clutch Timeline .....	69
Using External Gear Shifting (Simulink, Driving Simulator, etc.) .....	70
Appendix: Optimal Control Method for Steering Control .....	71
Theory .....	71

Application.....	73
A Linear 2D Vehicle Model .....	74
Speed Sensitivity.....	75
References .....	75

CarSim and TruckSim include all controls normally provided by a driver: steering, braking, throttle, gear shifting, and clutch control. Each control has options for both open-loop and closed-loop operation. When run in open-loop control mode, the controls can be defined with built-in Configurable Functions, or imported from external software.

VS Math Models also include closed-loop controllers to provide steering, braking, etc. A closed-loop path-following driver model (DM) is routinely used to steer the vehicle to follow a given path. A closed-loop speed controller (SC) is used to provide appropriate throttle, braking, shifting, and possibly clutch control. The two controllers interact: DM steering is affected by speed, and SC target speed may be affected by an intended target path.

The DM and SC controllers use Configurable Functions to define targets for paths and speed. The target speed and lateral offset Configurable Functions (`SPEED_TARGET` and `LTARG`, respectively) used by DM are also available for controlling motions of moving objects to simulate traffic vehicles and other targets for ADAS sensors. SC can also preview the path and combine the curvature with a target speed function and acceleration limits to provide realistic behavior for dynamic speed conditions.

In many cases, scenarios are simulated in which the control options change through VS Events. This can involve changing open-loop control datasets, or changing closed-loop target datasets, or even switching back and forth between open-loop and closed-loop control.

## Configurable Functions

Each open-loop control is represented in a VS Math Model with a Configurable Function that can be set to use one of many calculation methods. For example, Figure 1 shows the library screen for open-loop steering wheel angle control. In this case, the function type is set to **Linear interpolation, flat-line extrapolation** ①, and is based on a table of values of steering wheel angle vs. time ②. The nonlinear relationship from the table of numbers is shown graphically on the screen.

The closed-loop controllers also use Configurable Function to define target paths and target speeds.

## Documentation for Configurable Functions

There are many options for configuring these functions. Three reference manuals cover different aspects:

1. *VehicleSim Browser Reference Manual* describes the user interface for configuring the functions with screens such as the one shown in Figure 1,
2. *VS Math Models Reference Manual* lists all the built-in calculation methods and the associated keywords, and

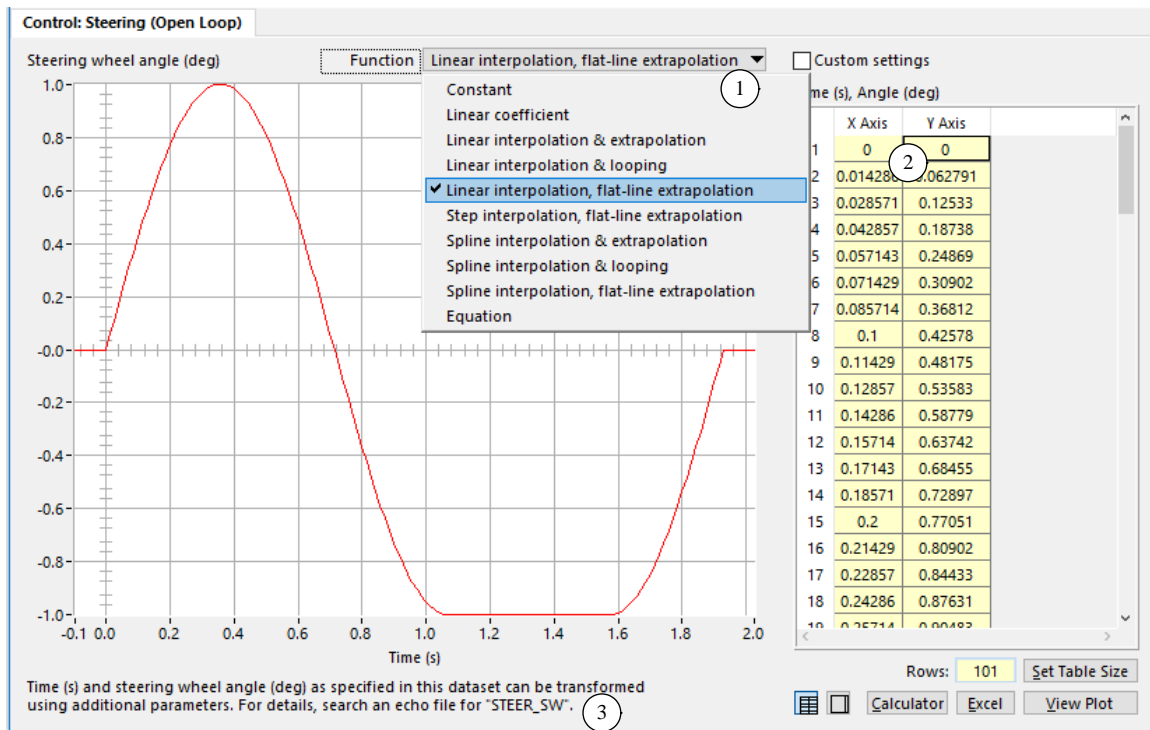


Figure 1. Open-loop steering control screen.

3. *VS Commands Manual* goes into more detail about extending the model with the `_EQUATION` option (to provide an equation for the function at runtime) and other methods for extending the model to use alternate equations.

Figure 2 shows part of an Echo file written at the end of a simulation run in which the `STEER_SW` waveform from Figure 1 was used and rescaled during the run. The comment lines at the top of the listing describe the options that are supported for this specific configurable function (constant, linear coefficient, nonlinear function of time, custom equation, etc.). The options are not the same for all configurable functions; the Echo file provides the reference to indicate which options can be used for each specific function.

After the documentation, the Echo file lists the data associated with the function. The type of function selected on the screen (1) (Figure 1) is defined in the parsfile with keywords `STEER_SW_TABLE` and `LINEAR_FLAT`, followed by a table of numbers that match the numbers shown on the screen (2).

After the keyword `END_TABLE`, four parameters (two scale factors and two offsets) are listed that can transform the shape of the function as described below.

```

ConTEXT - [Z:\2019.1 Dev\CarSim_Data\Results\Run_675e64f3-4bf5-42a2-95e1-fc894a958c9c\Last...
File Edit View Format Project Tools Options Window Help

! STEER_SW: Open loop steering wheel angle as a function of time. Steering wheel
! angle is a function of time (CONSTANT, COEFFICIENT, or TABLE). Alternatively, a
! custom equation can be defined at runtime. Steering wheel angle from the
! calculation can be adjusted with STEER_SW_GAIN and STEER_SW_OFFSET. Time used in
! the calculation can be adjusted with TSCALE_STEER and TSTART_STEER.

! 1D table: col 1 = time (s), col 2 = steering wheel angle (deg)
STEER_SW_TABLE LINEAR_FLAT ! linear interpolation, flat-line extrapolation
0, 0
0.014286, 0.062791
0.028571, 0.12533
0.042857, 0.18738
0.057143, 0.24869
0.071429, 0.30902
0.085714, 0.36812
0.1, 0.42578

< Edited to fit in figure >

1.8571, -0.30902
1.8714, -0.24869
1.8857, -0.18738
1.9, -0.12533
1.9143, -0.062791
1.9286, 0
ENDTABLE
STEER_SW_GAIN 112.357125 ! Gain multiplied with calculated value to get steering
! wheel angle
STEER_SW_OFFSET 0 ; deg ! Offset added (after gain) to get steering wheel angle
TSTART_STEER 43.9 ; s ! Offset subtracted from time
TSCALE_STEER 1 ! Scale factor divided into (time - TSTART_STEER)

Ln 4883, Col 1 Insert Sel: Normal Modified DOS File size: 305919

```

Figure 2. Part of Echo file showing Configurable Function for steering wheel angle (edited).

## Scaling reference waveforms

A built-in (native) open-loop control is calculated with a configurable function:

$$\begin{aligned}
 \text{native\_control} &= f(X) \cdot \text{gain} + \text{offset} \\
 \text{native\_control} &= f([t - tstart]/tscale) \cdot \text{gain} + \text{offset}
 \end{aligned}
 \tag{1}$$

where:

1.  $f$  is a function (e.g., STEER\_SW) that uses a method you define on the screen (e.g., linear interpolation);
2.  $X$  is the independent variable shown on the screen and used to calculate  $f$ ;  $X$  is in turn defined as a function of time and two parameters:  $X = [t - tstart]/tscale$ , where  $t$  is the simulation time,  $tstart$  is a parameter to offset the timeline of the control (e.g., TSTART\_STEER), and  $tscale$  is a parameter to scale the timeline of the control (e.g., TSCALE\_STEER);
3.  $gain$  is a dimensionless multiplier applied to the function (e.g., STEER\_SW\_GAIN); and
4.  $offset$  is an offset applied to the function (e.g., STEER\_SW\_OFFSET).

Notice that the table of numbers in the data screen shows the waveform starting at a time of zero, with a range of steering wheel angle limited to  $\pm 1^\circ$ . The amplitude of the waveform shown in the

figure is increased using the keyword `STEER_SW_GAIN`. Thus, the amplitude of the sine with dwell applied at the time the Echo file was written was not  $1^\circ$  as shown in the figure, but  $112.357^\circ$  (Figure 2). The waveform is delayed to start at  $T = 43.9\text{s}$  using the keyword `TSTART_STEER`. Thus, the waveform did not start at  $T = 0$  as shown in the figure, but at  $43.9\text{s}$ .

The other two parameters (`TSCALE_STEER` and `STEER_SW_OFFSET`) were not set in this run and keep the default values of 1 (`TSCALE_STEER`) and 0 (`STEER_SW_OFFSET`).

When a reference waveform is rescaled, the values of parameters are usually set on a different screen than the one that has the waveform. Typically, the scaling parameters are from the **Procedures** screen, the **Events** screen, or for quick changes, the **Run Control** screen. (Several examples are shown later.) As always, if the same parameter is specified in different datasets, the last value read by the VS Math Model overrides any previous value of the same function or parameter.

## Working with Multiple Datasets for Configurable Functions

There is an important distinction between the libraries used to specify path or speed and most of the other libraries in CarSim and TruckSim. Most libraries have properties for features that already exist in the VS Math Model (tires, steering system, options for control, etc.). For example, if your setup for a simulation run has two links for a steering system, the one that is received by the VS Math Model last will override properties specified in the dataset from the first link; parameter values that are read last are the ones used in the simulation.

VS Math Models support multiple reference paths and roads, defined at runtime with the commands `DEFINE_PATHS` and `DEFINE_ROADS`. The datasets for defining paths can add a new path and/or road with the appropriate command, and then set the properties of that path or road. If your setup has four links to paths, then four paths might be added to the model.

Applications for additional paths include guiding traffic vehicles, simulating lane detection, or other cases in which an S-L coordinate system is useful.

The same support for multiple Configuration Functions exists for the datasets that define target speed as a function of time, path station, or both via the `SPEED_TARGET` Configuration Function.

Paths, roads, `LTARG`, and `SPEED_TARGET` datasets have user-defined ID numbers. These allow the same ID number to be used for a path, road, lateral target, or speed target, regardless of when it was defined in a simulation. That is, a path with ID 1001 will have the same ID number whether it is the only path in the simulation, or whether it was defined fifth out of 15 paths.

The library screens that define tables for path, `LTARG`, or `SPEED_TARGET` all offer the options for setting the ID automatically or using a custom ID. If the intent for using the dataset is that will be the only one used in a simulation, then the automatic ID option is acceptable and possibly more convenient to use. However, if the dataset will be used in a scenario in which multiple datasets are needed, the custom ID numbers might be preferable.

There are two kinds of simulation where multiple datasets for path or speed Configuration Functions will be needed, and custom IDs should be used:

1. Simulations with multiple moving objects used to represent traffic vehicles and/or pedestrians. It is possible that each moving object will have position and motion specified

with `path`, `LTARG`, and `SPEED_TARGET` datasets. If the same `path`, `LTARG`, or `SPEED_TARGET` dataset will be applied to multiple moving objects, the simulation is less complicated if the same datasets are reused by specifying the same custom ID for the objects that use them.

2. Simulations with scenarios in which multiple paths and/or speeds are used for different parts of the scenario, typically changed using VS Events. If the scenario involves cycles where the same Event might be used multiple times, it is preferable to use custom IDs. The alternative will introduce a new Configurable Function dataset every time an Event is triggered that links to a `path`, `LTARG`, or `SPEED_TARGET` dataset. In lengthy simulations where Events might be triggered hundreds of times, it is possible that the VS Math Model will stop if a limit is reached for paths (the limit is 500 datasets), `LTARG` (the limit is 500), or `SPEED_TARGET` (the limit is 200).

## Open-Loop Controls

Open-loop controls that are defined in a VS Math Model are not influenced by any controllers built into the VS Solver libraries. When operating in open-loop mode, control variables are provided by Configurable Functions. The VS Math Models also support several options for settings the control variables from external software or user-defined equations, as described below.

The open-loop throttle and braking controls remain active when the built-in speed controller is used, in support of simulation of advanced safety systems that intervene with driver controls. Therefore, if you create a procedure that switches from open-loop to closed-loop mode during the run (using VS Commands and Events), you might need to set the open-loop controls to zero to prevent unwanted interactions.

## Replacing Open-Loop Variables with Custom Controller Outputs

Controls that are defined in this document as open loop can be set with closed-loop controllers defined with external software such as Simulink, or with VS Commands that are processed at runtime. They may also be replaced with values from a human driver using a driving simulator with hardware transducers that measure the control variables (steering, throttle, etc.) and provide those measured variables as imports to the VS Math Model.

The *VS Commands Manual* describes several methods for extending configurable functions that apply to the open-loop controls used in the math models:

1. Events can be defined in which some of the parameters are redefined when a condition is reached in the simulation.
2. Equations can be defined to calculate existing parameters or variables every time step.
3. A configurable function can be set to use an equation that defines the calculated variable  $f$  in terms of any of the thousands of variables that exist in the vehicle model that have associated keywords.

In addition to the built-in options, external software such as Simulink and LabVIEW can be used to define variables that are imported into the VS Math Model where they are applied. Imported variables can be combined with values computed internally using four potential methods:

1. The imported variable can replace the internal value.
2. The imported variable can be added to the internal value.
3. The imported variable can be multiplied with the internal value.
4. The imported variable can be ignored.

See the *VS Math Models Reference Manual* and *VS Commands Manual* for more information about import variables.

## Summary of Open-Loop Control Functions and Variables

All library screens for open-loop controls are similar in layout to the one shown in Figure 1 (page 3). Table 1 summarizes the functions, as defined by the keywords used to specify the controls. The root keyword in the table is used for the table function ( $f$  in the above equations), the import variable identifies *import*. Each function also has four parameters for transforming the calculated value (e.g., STEER\_SW) or time, as described earlier. The table also lists the output variable used to plot the function values.

*Table 1. Summary of Configurable Functions for open-loop controls.*

Control Library Screen Name	Root Keyword	Import Variable	Output
Control: Braking MC Pressure (Open Loop)	PBK_CON	IMP_PCON_BK	Pbk_Con
Control: Braking Pedal Force (Open Loop)	F_BRAKE_PEDAL	IMP_FBK_PDL	F_Pedal
Control: Clutch (Open Loop)	CLUTCH_CONTROL	IMP_CLUTCH	ClutchTr
Control: Shifting (Open Loop)	GEAR_TRANS	IMP_GEAR_TRANS	GearStat
Control: Steering (Open Loop)	STEER_SW	IMP_STEER_SW	Steer_SW
Control: Steering Torque (Open Loop)	M_STR_IN	IMP_STEER_T_IN	M_SW
Control: Throttle (Open Loop)	THROTTLE_ENGINE	IMP_THROTTLE_ENGINE	Throttle

**Notes** If multiple vehicles are defined in the VS Math Model, the import and output names have a suffix for vehicles 2 and higher, e.g., Pbk\_Con\_2 for vehicle 2.

Be aware that throttle will be automatically adjusted if it is given a value outside the range of zero to one. If negative, it is set to zero, and if greater than one, it is set to one.

Table 2 lists the conditions under which the open-loop controls are active.

Table 2. Summary of control libraries and modes.

Library	When Active
Control: Braking MC Pressure (Open Loop)	$OPT\_BK\_PEDAL = 0$
Control: Braking Pedal Force (Open Loop)	$OPT\_BK\_PEDAL = 1$ or $2$
Control: Clutch (Open Loop)	$OPT\_CLUTCH\_MODE = 0$
Control: Shifting (Open Loop)	$MODE\_TRANS = 1$
Control: Steering (Open Loop)	$OPT\_STR\_BY\_TRQ = 0$ and $OPT\_DM = 0$
Control: Steering Torque (Open Loop)	$OPT\_STR\_BY\_TRQ = 1$ and $OPT\_DM = 0$
Control: Throttle (Open Loop)	$INSTALL\_POWERTRAIN > 0$

## Reference Paths

A VS Reference Path is a continuous line that exists in a horizontal plane with continuity in position and gradient. That is, there are no sharp corners. A path defines a set of two coordinates that are useful for describing locations near the path: station  $S$  (distance along the path) and lateral coordinate  $L$  (distance a point is from the path, measured on a line that intersects the point and the path, and is perpendicular to the path at the point of intersection (Figure 3).

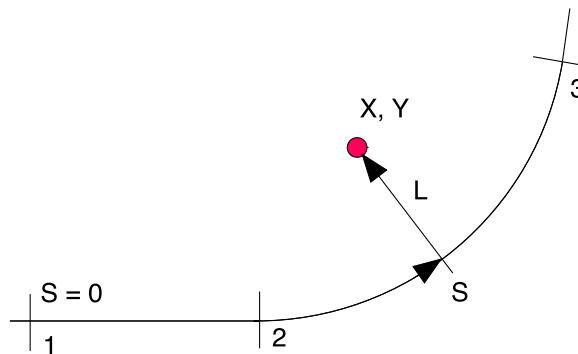


Figure 3.  $S$  and  $L$  are the coordinates of a reference path.

The main applications in VS Math Models for the  $S$  and  $L$  coordinates associated with a path are:

1. Closed-loop steering controllers work by trying to follow a target  $L$  defined as a function of  $S$  via the Configurable Function  $LTARG$ .
2. Moving objects may be located using reference paths and  $LTARG$  datasets.
3. Road elevation and friction are defined as functions of  $S$  and  $L$  for a road reference path if the ground is defined with VS Road surfaces. (Alternatively, elevation and friction are defined as functions of global  $X$  and  $Y$  coordinates if the ground is defined with a VS Terrain file.)

CarSim and TruckSim support up to 500 paths and include several means for defining them, as described in the document *Paths and Road Surfaces*. Each VS Road Surface requires a path as part of its definition. (However, surfaces defined with VS Terrain do not use paths.) In addition, multiple reference paths can be defined that are not associated with roads.



When paths are used to define targets for the closed-loop driver steering controller or to specify motions of moving objects, the **LTARG** Configurable Function is often used to specify the lateral target **L** as a function of **S** along a path. VS Math Models support up to 500 **LTARG** datasets.

The closed-loop DM controller for steer requires path and **LTARG** datasets, and the closed-loop speed controller (SC) has an option to use the same pair of path and **LTARG** datasets used by DM. The path used for path-following in DM may be part of a VS Road, or just exist in support of DM.

## Defining Reference Paths

CarSim and TruckSim include multiple libraries that are used to define reference paths (Table 3). These libraries can be found from the **Libraries** menu of the VS browser in the submenu **Paths and Road Surfaces**.

*Table 3. Libraries that define reference paths.*

Library Name	Description	VS Road?
Path: Segment Builder	Build a path from segments that are straight, pure arcs, curvature, tables of X-Y coordinates, and/or clothoids	No
Path: X-Y Coordinates for Segment	Build a path segment from X-Y table, used with the Path: Segment Builder	
Path: X-Y Coordinates	Build path with single X-Y table	
Path: X-Y Coordinates (Legacy)		
Path/Road: Segment Builder (Legacy)	Build a path and/or road from segments that are straight and arcs	Optional
Road: X-Y-Z Coordinates of Reference Line	Build path and road from X-Y-Z table	Yes
Road: X-Y-Z Coordinates of Edges	Build path and road from two X-Y-Z tables	
Scene: External Import	Import data from other tools	

The two libraries in Table 3 whose names begin with the word “Road” define a VS Road reference line along with some elevation information that varies with **S** and possibly both **S** and **L**. The library **Path/Road: Segment Builder (Legacy)** has options to determine whether it defines only a reference path or also includes some road properties such as banking for each segment. The libraries in Table 3 whose names begin with “Path” each define a reference path that can be used as input for DM or traffic vehicles, without being part of a road description. These paths may be used to define roads, but it is not required.

If a VS Terrain file is specified for a simulation run, then no VS Road surfaces are made, and paths are used only for providing controls for DM and moving objects.

The **Scene: External Import** screen is used to import path and possibly road data and possibly animator shape files from other sources. It is described in a separate document *Scene External Import* available from the **Help** menu (**Help > Paths, Road Surfaces, and Scenes**).

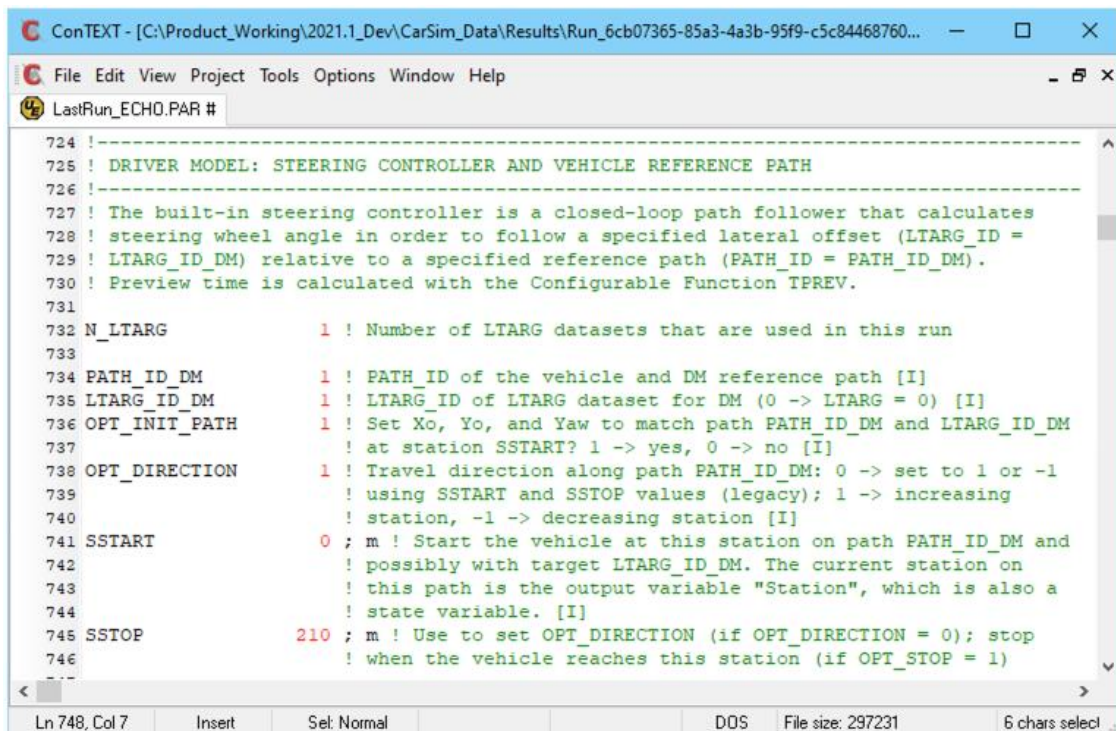
## Paths and LTARG Datasets Assigned to the Vehicle

The path and LTARG function associated with DM are also used to define the vehicle location.

### *The Driver Model part of the Echo file*

Figure 4 shows the top portion of the section in the Echo file that has information about the closed-loop steering controller.

The parameters listed in Figure 4 are available regardless of whether the driver model (DM) steering controller is used. At all times, a single reference path is assigned to the vehicle using the parameter PATH\_ID\_DM (line 734). If DM is active, this is the path used to control steering and possibly speed. Even if DM is not in use, the path whose ID matches PATH\_ID\_DM is used internally to define some path-related output variables such as Station and Lat\_Veh.



```
724 !-----
725 ! DRIVER MODEL: STEERING CONTROLLER AND VEHICLE REFERENCE PATH
726 !-----
727 ! The built-in steering controller is a closed-loop path follower that calculates
728 ! steering wheel angle in order to follow a specified lateral offset (LTARG_ID =
729 ! LTARG_ID_DM) relative to a specified reference path (PATH_ID = PATH_ID_DM).
730 ! Preview time is calculated with the Configurable Function TPREV.
731
732 N_LTARG          1 ! Number of LTARG datasets that are used in this run
733
734 PATH_ID_DM       1 ! PATH_ID of the vehicle and DM reference path [I]
735 LTARG_ID_DM      1 ! LTARG_ID of LTARG dataset for DM (0 -> LTARG = 0) [I]
736 OPT_INIT_PATH    1 ! Set Xo, Yo, and Yaw to match path PATH_ID_DM and LTARG_ID_DM
737                  ! at station SSTART? 1 -> yes, 0 -> no [I]
738 OPT_DIRECTION     1 ! Travel direction along path PATH_ID_DM: 0 -> set to 1 or -1
739                  ! using SSTART and SSTOP values (legacy); 1 -> increasing
740                  ! station, -1 -> decreasing station [I]
741 SSTART           0 ; m ! Start the vehicle at this station on path PATH_ID_DM and
742                  ! possibly with target LTARG_ID_DM. The current station on
743                  ! this path is the output variable "Station", which is also a
744                  ! state variable. [I]
745 SSTOP            210 ; m ! Use to set OPT_DIRECTION (if OPT_DIRECTION = 0); stop
746                  ! when the vehicle reaches this station (if OPT_STOP = 1)
```

Figure 4. Beginning of Driver Model section of Echo file, with vehicle initialization settings.

An LTARG dataset is optional and may be specified with the parameter LTARG\_ID\_DM (line 735).

In addition to setting output variables for the vehicle, PATH\_ID\_DM and LTARG\_ID\_DM may be used to set the initial location of the vehicle by setting the parameter OPT\_INIT\_PATH = 1 (line 736). In this case, the VS Math Model locates the vehicle at the specified starting station SSTART (line 741) for the driver reference path with a yaw angle parallel to the path. If LTARG\_ID\_DM > 0, then the location and yaw angle also take the specified LTARG target into account.

The vehicle is pointed in a direction along the path set by the parameter OPT\_DIRECTION (line 738). The SSTOP parameter is shown if it is needed to define a stop condition (when the System Parameter OPT\_STOP = 1).

The parameters `OPT_INIT_PATH`, `OPT_DIRECTION`, and `SSTART` are typically set from the **Procedures** library screen. The default value of `OPT_INIT_PATH` is 1. To disable this, uncheck the checkbox that sets `OPT_INIT_PATH` (Figure 5).

Figure 5. Section on Procedures screen for setting initialization details.

More details about the initializations are provided in the document *Paths and Road Surfaces*, available from the menu **Help > Paths, Road Surfaces, and Scenes**.

### Sequence of links

Each reference path dataset has a line of text that sets `PATH_ID_DM` to the path created by the dataset. Given that all reference path datasets will set `PATH_ID_DM` to the `PATH_ID` in the dataset, the vehicle `PATH_ID_DM` will be set to the last dataset scanned by the VS Math Model unless it is reset after that. In any simulation involving more than one path, it is a good idea to set `PATH_ID_DM` with a miscellaneous yellow field that is scanned after all of the Path datasets have been scanned.

## Changing Paths Using Events

VS Events are used to make changes to the VS Math Model during a simulation. Events are set up using datasets in the **Events** library, as described in document obtained from the menu item **Help > Procedures and Events**.

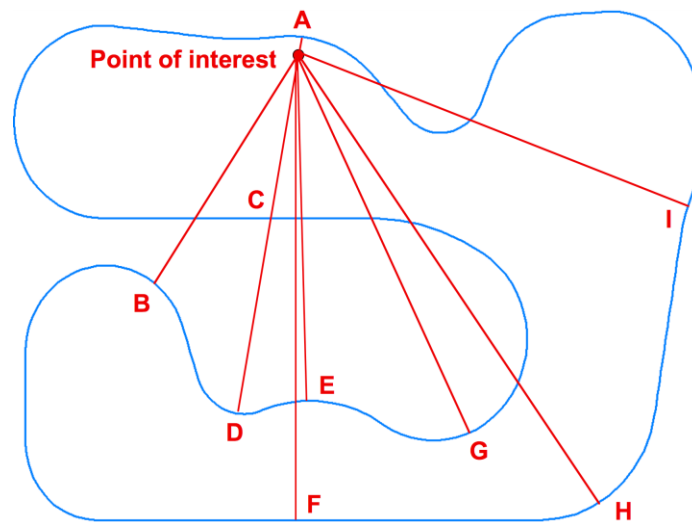
The **Events** library screen has a checkbox to specify initialization details with other checkboxes that are hidden by default (Figure 6). If the box is checked to **Specify initialization details** (1), the boxes for `OPT_INIT_PATH` (2), `OPT_INIT_SPEED` (3), and `OPT_INIT_CONFIG` (4) are initially unchecked (i.e., each parameter is set to 0). If a dataset from the **Events** library is loaded after a run starts, and these boxes are checked, the effect is that initializations will happen after the run has started. Usually, this is not wanted. However, the feature is sometimes used to simulate a sequence of tests in a single run, or to reset the vehicle position to begin a test maneuver.

Figure 6. Controls on the Event screen to re-initialize the vehicle.

Events can be set up to change the path used by the driver model during a run. This involves changing a single parameter: the path ID of the driver model: `PATH_ID_DM`. If this is the only change made, the vehicle state is not affected.

One complication that might occur with changing the path ID is that the definition of station on the new path might not match the station on the old path. Usually, the intent is for the vehicle state to remain “as is” with no change in position or speed of any moving part. In these cases, the VS Math Model must determine where the vehicle would be located along the new path by converting the global X and Y coordinate values of the vehicle to S and L path coordinates.

Converting from X-Y to S-L coordinates can sometimes be complicated for the VS Math Model. A major potential difficulty is that there might be many solutions of S-L coordinates that would be mathematically correct for a given vehicle location. For example, Figure 7 shows a path for a racecourse (blue) and a point of interest on the vehicle (red circle). Any of the points labeled A, B, etc. on the reference path are connected to the point with a line perpendicular to the path and are therefore valid locations for determining *S* and *L*. (In this case, point A is probably the intended location.)



*Figure 7. Multiple locations on a path might provide valid S-L coordinates for the vehicle.*

Internally, the VS Math Model keeps track of the most recent value of station for each point of interest. The station variable used by the closed-loop controllers for speed and steering is a state variable that can be accessed using its output name: *Station*.

If an Event is set up to change *PATH\_ID\_DM* during a run, it is usually a good idea to also set *Station* to a value that is close to where the vehicle will be located on the new path. For example, suppose the path shown in Figure 7 is activated during a run and the vehicle will enter near point A. The path station at point A is 2012 m, the path station at point C is 1243 m, and the current value of *Station* is 956 m. It is likely that the VS Math Model will either determine that the location on the path is point C ( $S = 1243$ ), or that it will fail to find a value and stop the run. To provide a clean transition, the parsfile loaded when the Event is triggered should not only load the new path dataset, it should also set *Station* to a value near 2000 to enable the VS Math Model to properly locate the vehicle’s position on the path.

Alternatively, the command *INIT\_CURRENT\_PATH\_SWEEP* can be used to allow the VS Math Model to attempt a guess based on closest proximity in the global coordinate system. Use this command in conjunction with assignment of a new value to *PATH\_ID\_DM*. The VS Math Model

will then search the entire path from end to end and assign the `Station` a value that is as close as possible to the vehicle's current X and Y global coordinate. This option may provide an automated solution to the problem outlined above at the cost of a slightly more expensive calculation.

**Note** When the `INIT_CURRENT_PATH_SWEEP` command is used, you cannot manually assign a value to `Station`. The option to explicitly set the value of `Station` is mutually exclusive with the `INIT_CURRENT_PATH_SWEEP` command.

## The Built-in Closed-loop Driver Model (DM)

CarSim and TruckSim include a built-in driver model (DM) controller that can be used to follow the path and target specified with the parameters `PATH_ID_DM` and `LTARG_ID_DM`.

Table 4 lists output variables related to DM. These variables exist in CarSim and TruckSim even if `OPT_DM` remains zero during the simulation.

*Table 4. Output variables associated with the DM controller.*

Short Name	Units	Long Name
Lat_Targ	m	Target lateral offset from path
Lat_Veh	m	Vehicle lateral distance to path
LtrgIdDm	-	ID of LTARG dataset used by DM
M_SW	N-m	Steering wheel torque
PathIdDm	-	ID of path dataset used by DM
PtchPath	deg	Pitch at ground on driver path
RhoPathY	1/m	Curvature of path (road lateral)
RhoPathZ	1/m	Curvature of path (road normal)
Rho_Road	1/m	Curvature of road X-Y ref. path
RollPath	deg	Roll at ground on driver path
Station	m	Station (path) at vehicle origin
Steer_DM	deg	Steering angle from DM controller
Steer_SW	deg	Steering wheel angle
X_Design	m	X coordinate of reference path
X_Target	m	X coordinate of target path
Y_Design	m	Y coordinate of reference path
Y_Target	m	Y coordinate of target path

**Note** If the VS Math Model has two or more vehicles, the short names shown in Table 4 have a suffix for vehicles 2 and higher that identify the vehicle, e.g., `Lat_Targ_2` for vehicle #2.

Figure 8 shows a continuation of the Echo file section that began in Figure 4 (page 10). This section begins with the parameter OPT\_DM (line 748) that can activate the driver model using one of three modes, indicated by values of 1, 2, or 3. In any of these cases, the model calculates the steering wheel angle Steer\_DM.

```

747
748 OPT_DM          3 ! Driver model option: 0 -> no driver model; 1 -> use linear
749                  ! dynamic model and 10 preview points; 2 -> same as 1, but
750                  ! with no rear steer effect (legacy); 3 -> use geometry and a
751                  ! single preview point [I]
752 OPT_DRIVER_ACTION 1 ! [D] Use steer from driver model (Steer_DM) when OPT_DM > 0?
753                  ! 1 -> use Steer_DM, 0 -> ignore Steer_DM [I]
754 OPT_STR_BY_TRQ    0 ! Control by steering wheel torque? 0 -> no, 1 -> yes [I]
755 A_SW_MAX_DM      540 ; deg ! Limit steering wheel angle for DM
756 AV_SW_MAX_DM     1200 ; deg/s ! Limit steering wheel rate for DM
757 VLOW_DM          10 ; km/h ! Minimum speed for preview dist = V*TPREV
758 ! XREF_DM         0 ; mm ! CALC -- Local X coordinate of DM reference point
759 XREF_DM_F         0 ; mm ! [D] X distance of DM ref. point in front of axle 1
760 XREF_DM_R         0 ; mm ! [D] X distance of DM ref. point in front of rear axle
761 YREF_DM          0 ; mm ! [D] Local Y coordinate of DM ref. point
762 INSTALL_DM_OUTPUTS ! VS Command to install XYZ outputs DM preview point(s)
763 ! NPREVIEW        0 ! No. of installed preview sensors for external DM (read only)

```

Figure 8. Parameters for driver model using single-point preview.

The parameter OPT\_DRIVER\_ACTION is available for advanced users to have DM calculate a steering wheel angle for use in another controller. When OPT\_DRIVER\_ACTION has the default value of 1, the steering wheel angle Steer\_SW is set to Steer\_DM. However, if OPT\_DRIVER\_ACTION = 0, the steer calculated by DM is not used; it is available for output and/or export to external models. The steering wheel angle may then be set using the import variable IMP\_STEER\_SW.

The parameters A\_SW\_MAX\_DM and AV\_SW\_MAX\_DM define limits on the range of Steer\_DM and the derivative of Steer\_DM, respectively.

The parameter VLOW\_DM defines the lowest speed where the controller behavior depends on vehicle speed. When the absolute value of the vehicle speed is less than this limit, the controller behaves as if the speed were VLOW\_DM. (It switches from a time preview to a distance preview, with the distance matching the preview at speed VLOW\_DM.)

## Steering by Torque

The parameter OPT\_STR\_BY\_TRQ determines whether the vehicle is controlled by torque at the steering wheel, or by the steering wheel angle. When OPT\_STR\_BY\_TRQ = 1 (steer by torque), a few more parameters are used in the driver model (Figure 9).



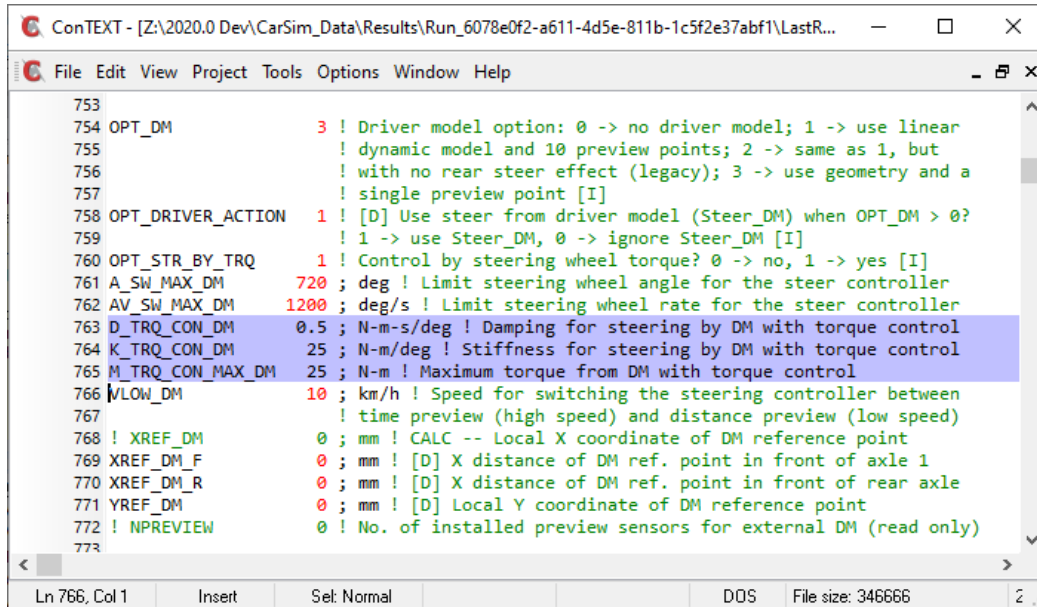


Figure 9. Additional parameters are used when steering by torque.

In this case, the desired steering wheel angle  $Steer\_DM$  is calculated. The difference between  $Steer\_DM$  and  $Steer\_SW$  is used with a spring ( $K\_TRQ\_CON\_DM$ ) and damper ( $D\_TRQ\_DM$ ) to apply steering wheel torque  $M\_SW$ , subject to a maximum limit  $M\_TRQ\_CON\_MAX\_DM$ .

The parameter  $OPT\_DRIVER\_ACTION$  has a similar effect when steering by torque: When  $OPT\_DRIVER\_ACTION$  has the default value of 1, the steering is set based on  $Steer\_DM$  and the spring/damping parameters as described above. However, if  $OPT\_DRIVER\_ACTION = 0$ , the steer calculated by DM is not used; it is available for output and/or export to external models. The steering wheel torque may then be set using the import variable  $IMP\_STEER\_T\_IN$ .

The combinations of  $OPT\_DM$ ,  $OPT\_STR\_BY\_TRQ$  and  $OPT\_DRIVER\_ACTION$  are summarized in Table 5.

Notice that the steer by torque option applies for both closed-loop and open-loop modes. Although the parameter  $OPT\_STR\_BY\_TRQ$  is a DM parameter, it does affect the behavior in the steering system model allowing it to accept torque controls. This involves adding degrees of freedom (DOFs) to the model to calculate steering rate and angle by integrating ordinary differential equations (ODEs). For more information, please see the *Steering Systems* document.

## DM Coordinate System

The DM axis system is shown in Figure 10 for a vehicle moving forward, along with the DM reference path and DM target path. The controller is set up to calculate steer to try to match the lateral position of the DM vehicle reference point (shown in red) relative to the reference path (the output variable  $Lat\_Veh$ ) with a target path specified using an  $LTARG$  dataset (the output variable  $Lat\_Targ$ ).

Table 5. Summary of DM options.

	OPT_STR_BY_TRQ = 0	OPT_STR_BY_TRQ = 1
OPT_DM = 0	<b>Open-loop SW angle control.</b> IMP_STEER_SW can be used to modify value from lookup table.	<b>Open-loop SW torque control.</b> IMP_STEER_T_IN can be used to modify value from lookup table.
OPT_DM > 0	<b>Closed-loop SW angle control.</b> <p>If OPT_DRIVER_ACTION = 1, then steering wheel angle is set to Steer_DM. IMP_STEER_SW can be used to modify the value from the controller.</p> <p>If OPT_DRIVER_ACTION = 0, then steering wheel angle is to be imported using IMP_STEER_SW. The output Steer_DM is calculated and available for potential use by an external controller.</p>	<b>Closed-loop SW torque control.</b> <p>If OPT_DRIVER_ACTION = 1, then the torque controller calculates an equivalent angle from Steer_DM and the spring/damper parameters which is then applied to the steering wheel. IMP_STEER_SW can be used to modify the value from the controller.</p> <p>If OPT_DRIVER_ACTION = 0, then steering wheel torque is to be imported using IMP_STEER_T_IN. The output Steer_DM is calculated for potential use by an external controller.</p>

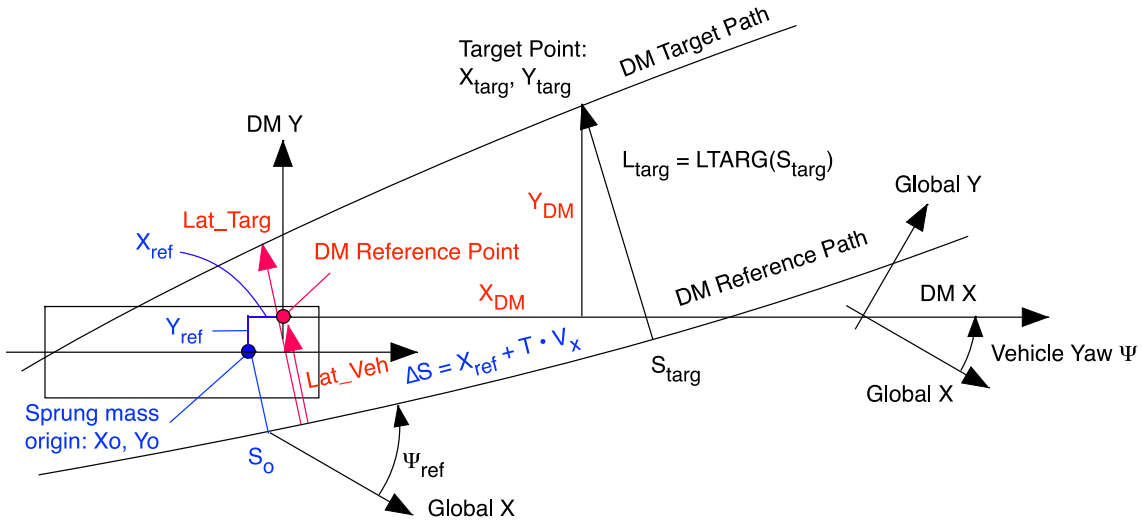


Figure 10. Axis system of steering controller and one preview point.

In the view of DM, the vehicle is located such that the origin of the vehicle sprung mass (shown in blue) is at local X = 0, local Y = 0. The yaw of the vehicle is defined as zero, such that the local X and Y axes are aligned with the longitudinal and lateral axes of the vehicle, which are rotated from the inertial axes by the vehicle yaw angle  $\psi$ .

Two alternative methods are available to calculate steer control. One uses simple geometry (as shown in the figure) and the other uses optimal control theory with a linear dynamic model.



## Steer Control Using Geometry and a Single Preview Point

When  $\text{OPT\_DM} = 3$ , a single preview point is used, as shown in the figure.

The controller considers a target point defined using preview distance  $\Delta S$  along the reference path. With the current station for the vehicle origin point along the reference path being  $S_o$ , the station used to determine the target point is:

$$S_{\text{targ}} = S_o + \Delta S \quad (2)$$

When the vehicle is moving forward ( $V_x > 0$ ), the road steer for the front wheels is calculated to point at the direction of the preview point. The specified preview is applied directly to the path station using a station increment  $\Delta S$ :

$$\Delta S = (X_{\text{ref\_f}} + T \cdot V_x) \cdot \text{OPT\_DIRECTION} \quad (3)$$

where  $X_{\text{ref\_f}}$  is the local X coordinate of the DM reference point,  $T$  is the preview time,  $V_x$  is the longitudinal vehicle speed, and  $\text{OPT\_DIRECTION}$  is the parameter with a value of  $\pm 1$  that determines which direction the vehicle is facing along the path.

The  $\text{LTARG}$  function is used to obtain the corresponding L coordinate for the target point,  $L_{\text{targ}}$ . A  $\text{VS}$  API function is used to obtain the global coordinates  $X_{\text{targ}}$  and  $Y_{\text{targ}}$  of a target point, given the path coordinates  $S_{\text{targ}}$  and  $L_{\text{targ}}$ .

The lateral position of the vehicle is evaluated at the DM reference point, shown in red in the figure.

The global coordinates  $X_{\text{targ}}$  and  $Y_{\text{targ}}$  are converted to local coordinates for the driver model  $X_{\text{DM}}$  and  $Y_{\text{DM}}$ :

$$X_{\text{DM}} = (X_{\text{targ}} - X_o) \cos(\Psi) + (Y_{\text{targ}} - Y_o) \sin(\Psi) - X_{\text{ref}} \quad (4)$$

$$Y_{\text{DM}} = (Y_{\text{targ}} - Y_o) \cos(\Psi) - (X_{\text{targ}} - X_o) \sin(\Psi) \quad (5)$$

where  $\Psi$  is the vehicle yaw angle.

When traveling forward ( $V_x > 0$ ), the angle going from the DM reference point to the target point is the steering control  $u_c$ :

$$u_c = \text{atan2}(Y_{\text{DM}}, X_{\text{DM}}) \quad (6)$$

When driving backwards, the reference point in the vehicle is relative to the rear axle, located at distance  $\text{LX\_AXLE}(2)$  behind the vehicle coordinate system origin for a 2-axle vehicle. In this case, the station increment  $\Delta S$  is:

$$\Delta S = (-\text{LX\_AXLE}(2) + X_{\text{ref\_r}} + T \cdot V_x) \cdot \text{OPT\_DIRECTION} \quad (7)$$

Steering is also reversed:

$$u_c = -\text{atan2}(Y_{\text{DM}}, X_{\text{DM}}) \pm \pi \quad (8)$$

where the sign of  $\pi$  has the same sign as  $\text{atan2}(Y_{\text{DM}}, X_{\text{DM}})$ .

The control angle is multiplied by an effective gear ratio to obtain the steering wheel angle from DM:

$$\text{Steer\_DM} = u_c \cdot \text{ratio} \quad (9)$$

where `ratio` is calculated dynamically in the model using the kinematical tables relating steering wheel angle to road wheel angle.

The global X, Y, and Z coordinates of the target point may be provided as output by using the command `INSTALL_DM_OUTPUTS` (line 758, Figure 8, page 14). If `OPT_DM = 3` when the command is read by the VS Math Model, the three outputs are created with the names `X_DM_1`, `Y_DM_1`, and `Z_DM_1`. Also, two import variables are added: `IMP_X_DM` and `IMP_Y_DM`. This allows the target point to be provided by external software or VS Commands.

The command `INSTALL_DM_IMPORTS` can be used to just load the two import variables `IMP_X_DM` and `IMP_Y_DM`. The import variables are loaded unconditionally, regardless of the status of `OPT_DM`.

The reference point has a local Y coordinate specified by the parameter `YREF_DM`, and an X coordinate that is forward of the center of the front suspension by the amount specified with the parameter `XREF_DM_F`, when travelling forward. When traveling backwards, the reference point is forward of the center of the rear suspension by the amount specified with the parameter `XREF_DM_R`. For example, Figure 11 shows part of an Echo file for a TruckSim simulation where the driver reference point is located at the outside rim of a front wheel, as required in a low-speed turning maneuver specified in a Performance Based Standards (PBS) Scheme.

```

1589      ! time preview (high speed) and distance preview (low speed)
1590      ! XREF_DM      0 ; mm ! CALC -- Local X coordinate of DM reference point
1591      XREF_DM_F      0 ; mm ! [D] X distance of DM ref. point in front of axle 1
1592      XREF_DM_R      0 ; mm ! [D] X distance of DM ref. point in front of rear axle
1593      YREF_DM      -1215 ; mm ! Local Y coordinate of DM reference point
1594      INSTALL_DM_OUTPUTS      ! VS Command to install XYZ outputs DM preview point(s)
1595      ! NPREVIEW      0 ! No. of installed preview sensors for external DM (read only)
1596

```

Figure 11. Example with driver reference point at right-front wheel rim.

Figure 12 shows results from the simulation. Notice the sphere shown on the dashed yellow line on the outside of the right-front wheel, preceded by another sphere about a meter in front.

In this example, the Y coordinate `YREF_DM` is not only nonzero; it is also updated dynamically with VS Commands to shift based on an integral of the tracking error. As shown at the end of the Echo file (Figure 13), new output variables were added for this example for the lateral error `LatError` (line 9940) and the integral of the lateral error, `IntLerr` (lines 9941 and 9969). The lateral coordinate of the driver model reference point is set to half the front track wide plus the distance to the tire reference point and adjusted by the integral of the tracking error (line 9959).

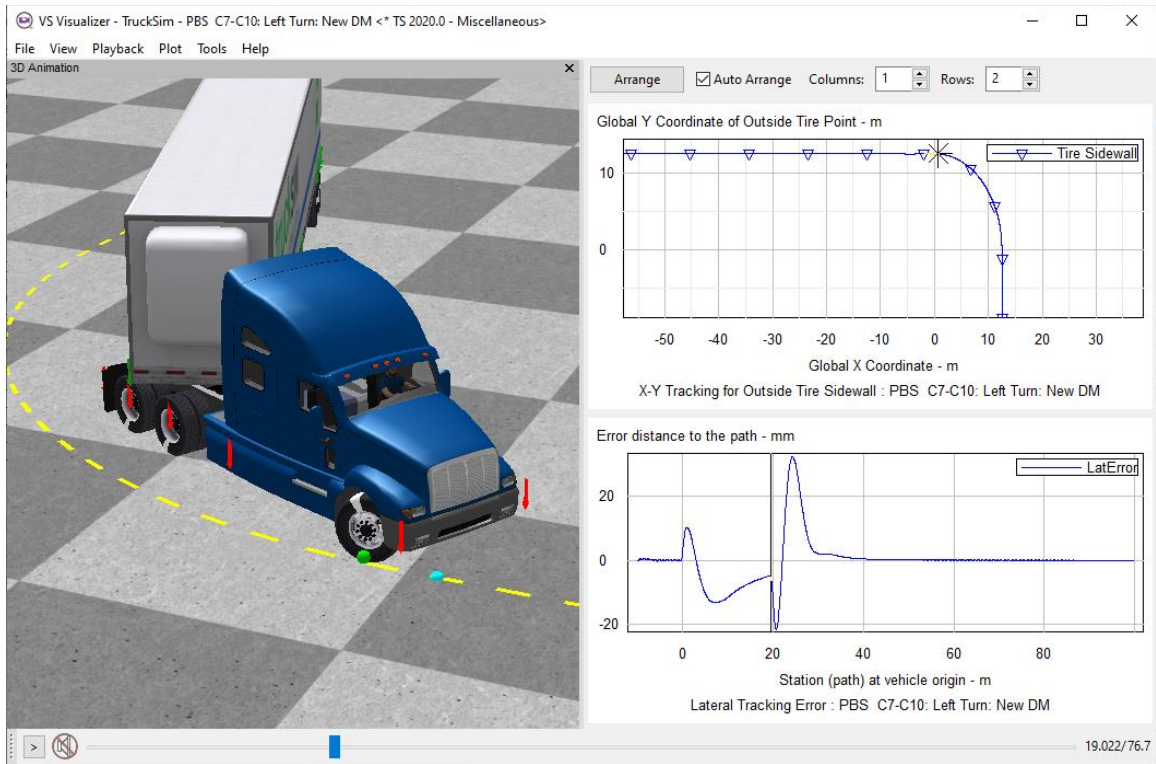


Figure 12. TruckSim example with reference point located on the outside of the right-front wheel.

```

9940 DEFINE_OUTPUT LatError = 0; mm ; Error distance to the path
9941 DEFINE_OUTPUT IntLerr = 0; - ; Integral of LatError
9942 DEFINE_OUTPUT LatFront = 5; mm ; Outer corner tracking
9943 DEFINE_OUTPUT FricL1 = 0; - ; FricL1
9944 DEFINE_OUTPUT FricR1 = 0; - ; FricR1
9945 DEFINE_OUTPUT FricTot1 = 0; - ; FricTot1
9946
9947 !-----
9948 ! INITIALIZATION EQUATIONS (APPLIED JUST AFTER INITIALIZATION)
9949 !-----
9950 EQ_INIT ROAD_ID_OBJ(1) = CURRENT_ROAD_ID;
9951
9952 !-----
9953 ! EQUATIONS OUT (AT THE END OF EVERY TIME STEP)
9954 !-----
9955 EQ_OUT X_REF = XCTC_R10 + SIN(YAW_R1)*L_TIRE_REF;
9956 EQ_OUT Y_REF = YCTC_R10 -(COS(YAW_R1)*L_TIRE_REF);
9957 EQ_OUT LATERROR = PATH_L_I(X_REF, Y_REF, 2);
9958 EQ_OUT LATFRONT = -PATH_L_I(X_S6, Y_S6, 2);
9959 EQ_OUT YREF_DM = -(L_TIRE_REF + L_TRACK(1,1)/2) + INTLERR;
9960 EQ_OUT FRICL1 = 100*SQRT(FX_L1^2 + FY_L1^2)/FZ_L1;
9961 EQ_OUT FRICR1 = 100*SQRT(FX_R1^2 + FY_R1^2)/FZ_R1;
9962 EQ_OUT FRICTOT1 = 100*(SQRT(FX_L1^2 + FY_L1^2) + SQRT(FX_R1^2 + FY_R1^2))/(FZ_L1 + FZ_R1);
9963 EQ_OUT X_OBJ_1 = X_DM_1;
9964 EQ_OUT Y_OBJ_1 = Y_DM_1;
9965
9966 !-----
9967 ! DIFFERENTIAL EQUATIONS FOR NEW STATE VARIABLES (AT THE END OF EVERY TIME STEP)
9968 !-----
9969 EQ_DIFFERENTIAL INTLERR = LATERROR;

```

Figure 13. Echo file showing equation to update YREF\_DM each time step.

**Note** The parameters XREF\_DM\_F, XREF\_DM\_R, and YREF\_DM were added in version 2020.0 and are intended for advanced users. They are not

supported in any fixed fields in the GUI; they are typically set from a miscellaneous yellow field, such as the one on the screen **Control: Steering by the Closed-Loop Driver Model** (see page 25).

The default values of the parameters  $XREF\_DM\_F$ ,  $XREF\_DM\_R$ , and  $YREF\_DM$  are all zero, so the reference point is usually at the center of the front suspension when moving forward, or the center of the rear suspension when moving in reverse.

## Steer Control using Linear Dynamics and Multiple Preview Points

When  $OPT\_DM = 1$  or  $2$ , a method based on optimal control of a linear dynamic system is used that was developed by Charles MacAdam of UMTRI in 1980 [1, 2]. The algorithm presented in this document has been streamlined and re-formulated to work with reference paths for use in CarSim and TruckSim. Over time, the controller has been extended to account for suspension steer effects (front and rear) and active rear steering. It has also been extended to support driving in reverse.

The general control method is programmed to generate a steering wheel angle in a vehicle model. The algorithm flow is shown in Figure 14. Given a target path, the steering controller computes a steering wheel angle given the current state of the vehicle.

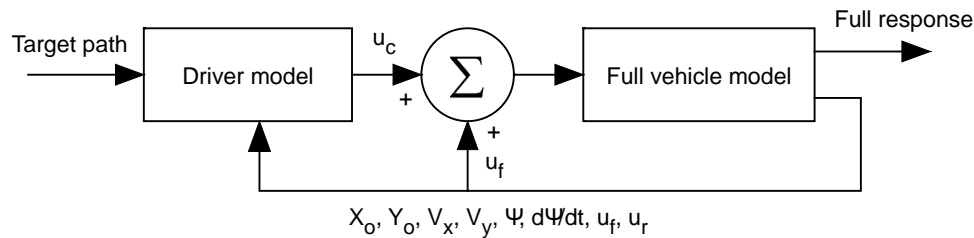


Figure 14. Algorithm flow.

The algorithm calculates the optimal control  $u$  to minimize deviations of the path of a vehicle reference point from a target path and subtracts front and rear steering ( $u_f$  and  $u_r$ ) from other sources (kinematics and compliance) to obtain the steering needed by the driver,  $u_c$ .

The appendix (page 68) provides the theory and application of this method in the steering controller.

The algorithm assumes multiple preview points spaced evenly in the direction of the vehicle X axis (Figure 15).

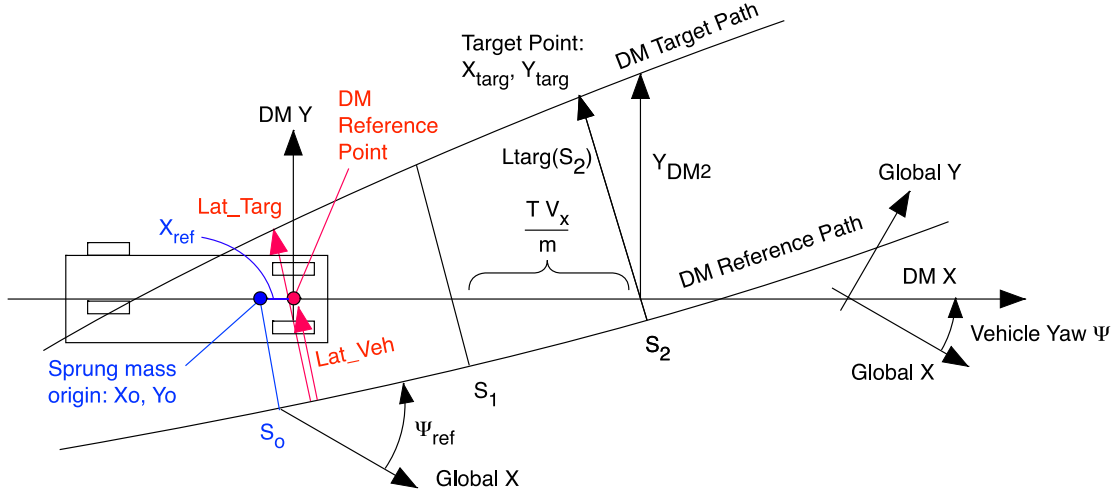


Figure 15. Geometry for multiple preview points.

When moving forward ( $V_x > 0$ ), each preview point has a local X coordinate:

$$X_{DM} = X_{ref} + \frac{i T V_x}{m} \quad (10)$$

where  $X_{ref}$  is the local X coordinate of the DM reference point,  $i$  is the interval number,  $T$  is the overall preview time, and  $m$  is the number of intervals. The corresponding station is approximately:

$$\Delta S = X_{DM} \cos(\Psi_{rel}) \quad (11)$$

where  $\Psi_{rel}$  (heading angle of vehicle relative to the reference path) is assumed to be constant over the preview distance. When the vehicle is pointed in the direction of the path for increasing station (the parameter  $OPT\_DIRECTION = +1$ ),  $\Psi_{rel}$  is the difference between the path heading angle  $\Psi_{ref}$  and the vehicle yaw angle  $\Psi$ . However, when the vehicle is pointed in the other direction along the path (towards decreasing station, with  $OPT\_DIRECTION = -1$ ), the difference between the two angles is adjusted by  $180^\circ$ .

Given the path station  $S_{targ}$ , the  $L_{TARG}$  function is used to obtain the corresponding  $L$  coordinate for the target,  $L_{targ}$ ; the path coordinates  $S_{targ}$  and  $L_{targ}$  are then converted to global coordinates  $X_{targ}$  and  $Y_{targ}$  for each target point. The local Y coordinate is then obtained using the vehicle yaw angle  $\Psi$  and the relative heading  $\Psi_{rel}$ :

$$Y_{DM} = [(Y_{targ} - Y_o) \cos(\psi) - (X_{targ} - X_o) \sin(\psi)] / \cos(\Psi_{rel}) \quad (12)$$

Equations 10 and 12 are based on the assumptions that the heading of the reference path is constant in the preview range, and that the target path and reference paths are parallel.

Ten preview points are used ( $m = 10$ ), and the road steer for the front wheels is the theoretical control  $u$  minus the steering due to suspension kinematics and compliance:

$$u_c = \frac{\sum_{i=1}^m [y_{targ,i} - F_i x_o - h_i v] g_i W_i}{\sum_{i=1}^m g_i^2 W_i} - u_f \quad (13)$$

Equation 13 is based on equation 32 in the Appendix, with variable  $y_{targ,i}$  in equation 13 corresponding to  $Y_{DM}$  in equation 12.

When the linear dynamic controller is used, two more parameters are shown in the Echo file (Figure 16). The parameter TLAG (line 745) is available when using the linear dynamic controller; it adds a pure time delay to Steer\_DM. This delay was used in old research to replicate the delays observed with human drivers. It is seldom used, and the recommended value is 0 unless there is a need for including the delay.

```

733
734 OPT_DM          1 ! Driver model option: 0 -> no driver model; 1 -> use linear
735                  ! dynamic model and 10 preview points; 2 -> same as 1, but
736                  ! with no rear steer effect (legacy); 3 -> use geometry and a
737                  ! single preview point [I]
738 OPT_DM_2019      1 ! Option: 0 -> use the documented equations, 1 -> use legacy
739                  ! equations (2019.0 and earlier)
740 OPT_DRIVER_ACTION 1 ! [D] Use steer from driver model (Steer_DM) when OPT_DM > 0?
741                  ! 1 -> use Steer_DM, 0 -> ignore Steer_DM [I]
742 OPT_STR_BY_TRQ    0 ! [D] Control by steering wheel torque? 0 -> no, 1 -> yes [I]
743 A_SW_MAX_DM      540 ; deg ! Limit steering wheel angle for the steer controller
744 AV_SW_MAX_DM     1200 ; deg/s ! Limit steering wheel rate for the steer controller
745 TLAG_DM          0 ; s ! [D] Lag time in steer controller [I]
746 VLOW_DM          10 ; km/h ! Speed for switching the steering controller between
747                  ! time preview (high speed) and distance preview (low speed)
748 ! C_F_DM 2595.613464 ; N/deg ! CALC -- total cornering stiffness, all front tires
749 ! C_R_DM 1531.994111 ; N/deg ! CALC -- total cornering stiffness, all rear tires
750 ! IZZ_DM 2768.093804 ; kg-m2 ! CALC -- yaw moment of inertia assumed by DM
751 ! LX.CG_DM 1059.249292 ; mm ! CALC -- distance between front axle(s) and mass center
752 ! LX.WB_DM 2910 ; mm ! CALC -- wheelbase
753 ! M_DM 1412 ; kg ! CALC -- mass assumed by DM (total static loads)
754 ! XREF_DM -0 ; mm ! CALC -- Local X coordinate of DM reference point
755
756 INSTALL_DM_OUTPUTS ! VS Command to install XYZ outputs DM preview point(s)
757 ! NPREVIEW 0 ! No. of installed preview sensors for external DM (read only)
---
```

Figure 16. DM parameters for the linear model with 10 preview points.

In version 2019.1, the driver model was revised extensively to include the single-preview controller described in the previous subsection, and to include control while driving backwards. In doing so, some revisions were made to the internal equations. The parameter OPT\_DM\_2019 provides a means to use the old (2019.0 and earlier) equations, in support of version-to-version testing. This parameter is not applicable if the vehicle drive mode is -1 (driving backwards) or if the single-point method is used (OPT\_DM = 3).

When set to zero (the default), the equations presented in this document are used. Also, when driving backwards, or using the single-point method (OPT\_DM = 3), the new equations are used, regardless of the value of the OPT\_DM\_2019 parameter. When there is a need to reproduce results from older versions, set OPT\_DM\_2019 = 1.

Here are the major differences between the old and new versions:

1. Equation 11 for  $\Delta S$  did not include the factor:  $\cos(\Psi_{rel})$ .
2. Equation 12 for  $Y_{DM}$  did not include the divisor:  $\cos(\Psi_{rel})$ .
3. The third elements of the B and H arrays (equation 34) were not used.
4. The effects of rear steer (active and suspension) were reduced by 50%.
5. The initial yaw angle of the vehicle was set to be parallel with the reference path, but did not include any angle associated with the specified LTARG offset.

Properties of the vehicle are used to calculate the coefficients of the matrices shown in the Appendix; these are provided in the Echo file as calculated properties (lines 748 – 753).



**Note** If the VS Math Model has two or more vehicles, the linear dynamics controller ( $OPT\_DM = 1$  or  $2$ ) may only be used with vehicle #1. However, the single-point controller ( $OPT\_DM = 3$ ) may be used with all vehicles in the VS Math Model.

## Viewing the DM Preview Point(s)

The DM Controller uses either one ( $OPT\_DM = 3$ ) or ten ( $OPT\_DM = 1$  or  $2$ ) preview point(s) to determine a steering wheel angle. A command `INSTALL_DM_OUTPUTS` is provided to enable advanced users to generate the global X, Y, and Z coordinates of the point(s). For each point  $i$  ( $i = 1, 2, \dots, 10$ ), the three output variables are named  $X\_DM\_i$ ,  $Y\_DM\_i$ , and  $Z\_DM\_i$ . Put the command into any miscellaneous yellow field to create these output variables.

Both CarSim and TruckSim include example datasets that shows one sphere for the single-point driver preview (Figure 17), and 10 spheres at the driver preview points with the older control options.

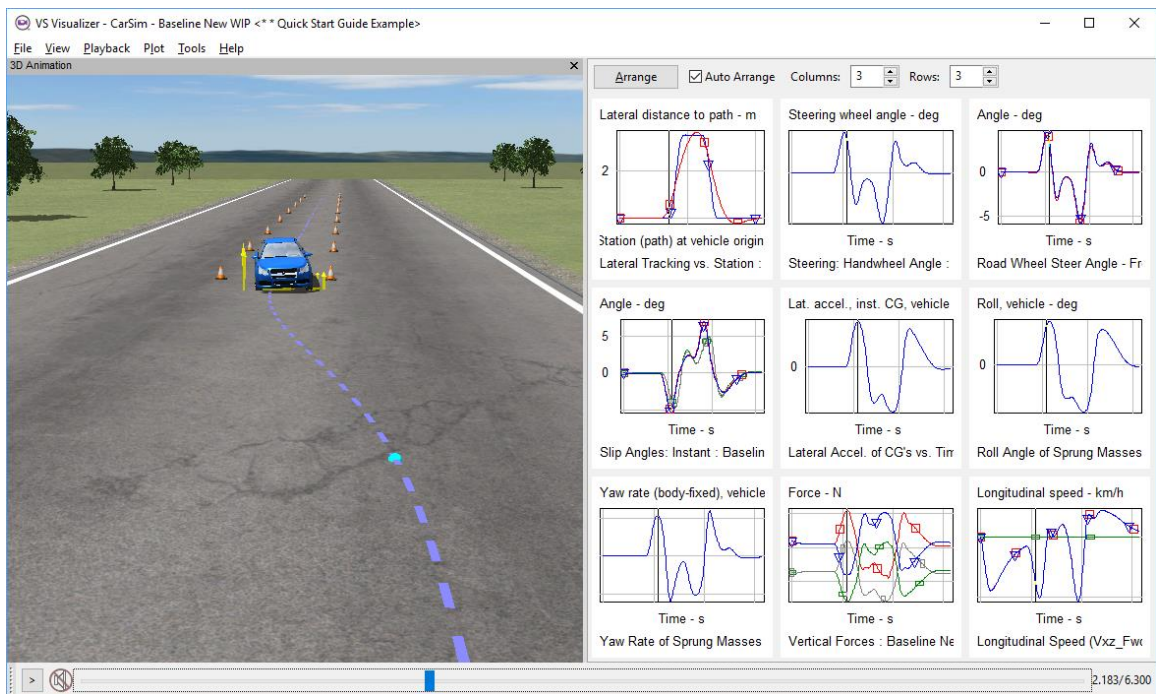


Figure 17. A single preview point for the driver model shown with an animated sphere.

The **Multiple Moving Objects** library contains several datasets using small spheres to show points of interest. Figure 18 shows the setup to support one driver preview point. Other datasets show more points of interest.

**Multiple Moving Objects**

VS Commands apply (1) each object before anything else set on this screen

INSTALL\_DM\_OUTPUTS ! Need outputs from the driver model  
EQ\_INIT ROAD\_ID\_OBJ(<o>) = CURRENT\_ROAD\_ID ! now (2)  
EQ\_OUT X\_Obj(<o>) = x\_dm\_1 ! Calculate every time step (3)  
EQ\_OUT Y\_Obj(<o>) = y\_dm\_1

Path ID: 0 (4) Road ID: 5 (5)

Do not specify station or lateral location here

1 ▾ Objects

Object Specifications

3D Shape for Video: Shape File ▾  
Sphere: 1m Radius (Scaled for 0.2m)

Set color? ☒  

☐ Recycle objects

☐ Sensor/Target Shape and Specifications

Figure 18. Example dataset with a moving object that shows the driver preview point.

In this example, the `INSTALL_DM_OUTPUTS` command is written into the miscellaneous yellow field (1). The Path ID is set to 0 (4), indicating that the horizontal location of each object is specified with global X and Y coordinates. Those coordinates are provided for each moving object with the VS Command `EQ_OUT` (3). The Z coordinate for each object will be calculated using the vehicle road with ID `CURRENT_ROAD_ID`. However, at the time this dataset is loaded, `CURRENT_ROAD_ID` might not be known. Therefore, the yellow field (5) is set to 0, and an `EQ_INIT` command is used to set the road ID of the object to `CURRENT_ROAD_ID` after the initialization is complete (2).

## Using DM with External Models

If the single-point option has been selected (`OPT_DM = 3`), two import variables are also created in support of advanced users calculating the X and Y coordinates of the preview point using VS Commands or external models (e.g., Simulink). Given that the driver model simply points the steered wheels at the point, the steering can be controlled by providing the X and Y coordinates of the preview point. The import variables have keywords `IMP_X_DM` and `IMP_Y_DM`.

Another option is to make use of the parameter `OPT_DRIVER_ACTION` (for advanced users, set by typing the keyword and value into a miscellaneous yellow field). Setting `OPT_DRIVER_ACTION = 1` (the default) causes the driver model to calculate a steering wheel angle and apply the calculated steering wheel angle to the steering system. Setting `OPT_DRIVER_ACTION = 0` causes the driver model to calculate a steering wheel angle for output purposes, but the calculated steering wheel angle is not applied to the steering system. The steering wheel angle calculated by the driver model is output in the variable `Steer_DM`. This can be used to provide a control input for an externally defined steering control system, such as a steer by wire. (Consider it a “steering request” from the driver.) The calculated driver control is available as an export for use in external software such as Simulink, or in equations defined with VS commands.

## The Auxiliary Steering Controller

Setting `OPT_DM_AUX` to 1 enables an Auxiliary Steering Controller. When enabled, this feature produces an output for a “bicycle model” road wheel steer angle, `StrCtlAux`, that follows an `LTARG` (linked to the run and whose ID is the value assigned to `LTARG_ID_AUX`) different from the one in use by the internal driver model. Other input parameters are `XREF_AUX_DM`, the distance ahead of the front axle of the reference control point on the vehicle and `YREF_AUX`, the



distance to the left of the vehicle centerline of the reference control point on the vehicle. Other parameters are shared with those set for the internal closed-loop driver model.

Note that by itself this control does nothing at all, except generate output data. The intended use of the feature is support for development of autonomous and ADAS controls, where the controller needs information about a target path offset different from the one in use by the internal driver model. Users must supply their own code using VS Commands, Simulink, or any of the other methods of extending the math model to make use of the data.

A few caveats apply. This controller assumes use of single point control for the driver model ( $\text{OPT\_DM} = 3$ ). Also, the internal driver model should be set to torque control ( $\text{OPT\_STR\_BY\_TRQ} = 1$ ). Otherwise, the action of the internal steer controller is to constrain the steering wheel to a position, regardless of other inputs applied to the system. When steered by torque, the internal driver model can be given an input torque of zero, and the steering column will follow the steering gear input angle.

## Trailer Backing Controller

The *trailer backing controller* uses the steering wheel angle to control the hitch articulation angle such that it matches the hitch angle needed for the trailer to follow the desired path. This is an optional feature building upon the single preview point mode ( $\text{OPT\_DM} = 3$ ). For more information, refer to the *Trailer Backing Controller* help file.

## Steering by the Closed-Loop Driver Model: GUI Screen

DM is usually configured using the library screen shown in Figure 19. The top part of the screen shows a lateral offset associated with the LTARG Configurable Function. The bottom part shows properties of the DM controller.

### Lateral Target

The controls at the top of the screen define an LTARG dataset for that can be used for DM or to specify motions of moving objects.

- ① Target relative to a reference path that calculates  $L$  as a function of station  $S$  along the path. As with most Configurable Functions, this can be a constant, a coefficient, or a nonlinear table specified with a set of numbers and a choice of interpolation and extrapolation methods as described in the *VehicleSim Browser Reference Manual* and *VS Math Models Reference Manual*.
- ② Drop-down control for setting the user ID for this LTARG dataset. This Configurable Function includes a user ID (keyword =  $\text{LTARG\_ID}$ ). The ID can be set automatically, or an ID can be specified that is 999 or higher. Details on setting and using ID numbers for Configurable Functions are described in the *VehicleSim Browser Reference Manual*.

If the ID is set automatically, the Parsfile for the dataset increments the number of LTARG datasets ( $\text{N\_LTARG}$ ) and assigns the new value of  $\text{N\_LTARG}$  to the ID,  $\text{LTARG\_ID}$ . For example, if there were already three LTARG datasets, the automatically set ID for a new dataset will be  $\text{LTARG\_ID} = 4$ .

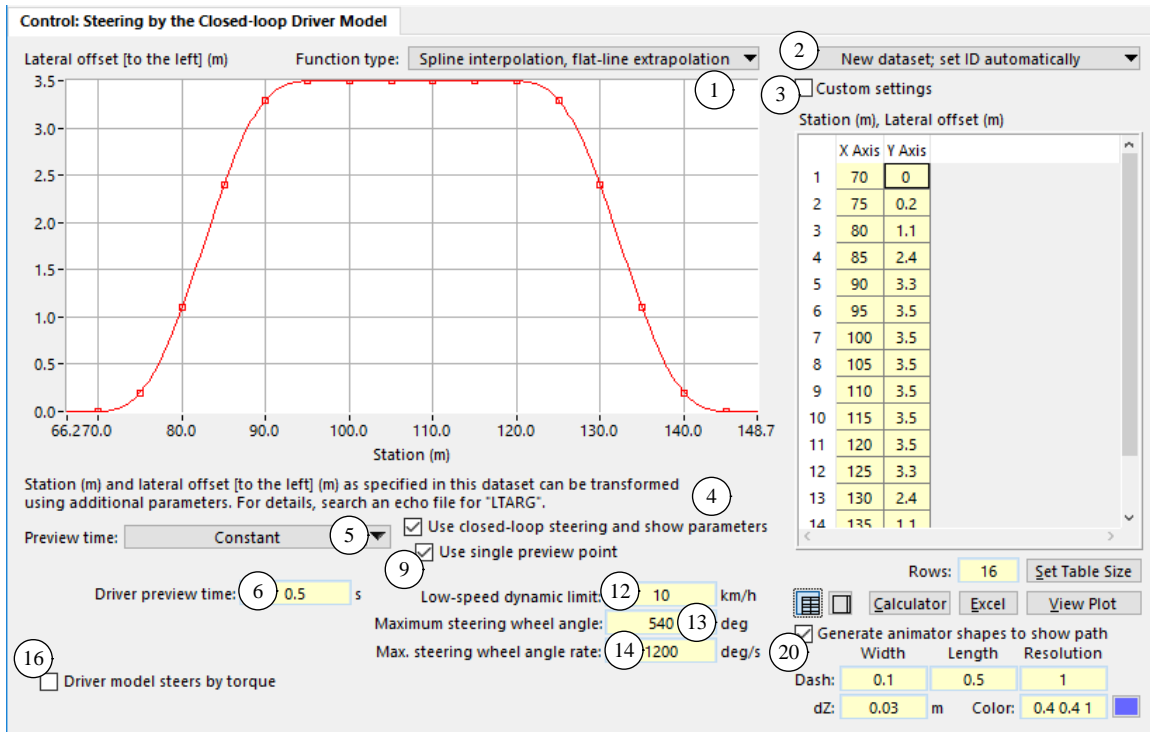


Figure 19. Library screen for steering by the closed-loop driver model.

As described earlier (page 5), the custom ID option is recommended if multiple LTARG datasets will be used in a simulation.

- ③ Checkbox to show a miscellaneous field for custom settings. Advanced users can enter other properties or VS Commands to extend the model.

## DM Settings

When the checkbox ④ is checked, this screen activates the DM controller, and provides the settings for DM.

- ④ Checkbox **Use closed-loop steering and show parameters**. If this box is not checked, then all the controls ⑤ and higher are not shown (everything in the lower portion of the screen). Uncheck this box if the intent is to create an LTARG dataset without affecting the built-in driver model. For example, the LTARG dataset may be used for controlling the positions of moving objects, as described in the **Help** document *ADAS Sensor and Moving Objects*.

When checked, the path follower controls are shown, and the parameter LTARG\_ID\_DM is set to the LTARG\_ID for the LTARG dataset.

If not checked, the Parsfile for the dataset has no references to any DM parameters

- ⑤ Control to set the preview time. A drop-down control is used to select between different options for defining the preview time (Figure 20).

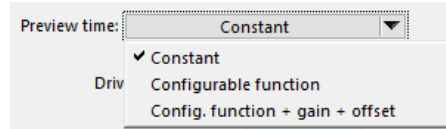


Figure 20. Options for defining preview time.

When the drop-down list is set to **Constant** (Figure 19) the preview time is constant as specified in a yellow field (6).

When the drop-down list is set to **Configurable function** or **Config. function + gain + offset** a link is displayed under the drop-down control (Figure 21). The linked dataset is for the Configurable Function named TPREV, which is used internally to calculate preview time T varying with speed. When the option is set to **Config. function + gain + offset**, additional yellow fields are shown for the offset (7) and gain (8).

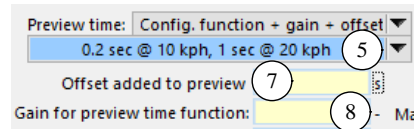


Figure 21. Appearance when a Configurable Function is selected.

- (6) Constant preview time (keyword = TPREV\_CONSTANT). This field is shown only when the drop-down control (5) is set to **Constant** (Figure 19).

The preview time is typically set somewhere between 0.1 and 2.0 seconds. Some considerations:

1. If a single preview point is used (checkbox (9) is checked), the preview time is typically set smaller than when the preview time covers ten intervals (checkbox (9) is not checked).
2. Longer preview times give more stable behavior, with less tracking accuracy.
3. Short preview times give more tracking accuracy with potential loss of stability if checkbox (9) is not checked.

At speeds below a specified limit (12), the preview is a fixed distance corresponding to the preview time that would apply at the low-speed limit.

- (7) Offset is added to preview time from table (keyword = TPREV\_OFFSET). This field is only visible when the drop-down list (5) is set to **Config. function + gain + offset**. Using this option, you can define a normalized table of a standard shape and adjust it using the gain and offset values. (See equation 1 on page 4.)
- (8) Gain for the speed-varying driver preview time (database keyword = TPREV\_GAIN). This field is only visible when the drop-down list (5) is set to **Config. function + gain + offset**. Use it to set a scale factor to multiply by the values from the table. Using this option, you can define a normalized table of a standard shape and adjust it using the gain and offset values. (See equation 1 on page 4.)

- ⑨ Checkbox to use a single preview point. This chooses between approaches for calculating steer angle.

1. When checked ( $OPT\_DM = 3$ ), the steering control is made to steer the front wheels with an angle parallel to a line that goes from the DM reference point to a point  $L_{prev} = T_{prev} \cdot |V_{fwd}|$  in front of the reference point, where  $T_{prev}$  is the preview time. That is, the steer is zero if the target point is directly in front of the reference point. The reference point is nominally at the center of the front suspension when driving forward and is nominally at the center of the rear suspension when driving backward. This controller was described earlier (page 16).
2. When not checked ( $OPT\_DM = 1$  or  $2$ ), the steering control is calculated using a built-in linear dynamic vehicle model to predict the steering that matches the target path for ten points covering the distance  $L_{prev}$  in front of DM reference point, as described earlier (page 19). In this case, more controls are shown: ⑩, ⑪, and ⑮ (Figure 22).

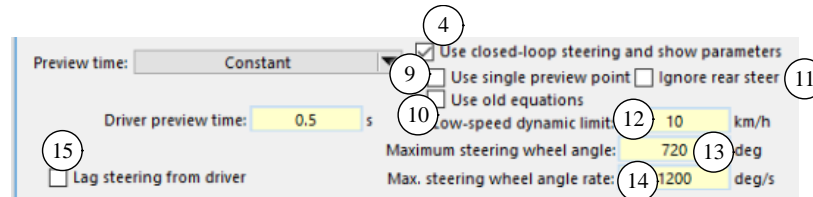


Figure 22. More options are shown with multiple preview points.

**Notes** The single-point method ( $OPT\_DM = 3$ ) was added to version 2019.1; the other two options have been available in previous versions for decades.

The single-point method is more robust and is recommended for most simulations, especially those involving limit conditions.

The preview time for a single point should typically be set to about half the value that would be used for ten points when this box is not checked.

- ⑩ Checkbox with an option to use old equations when the linear dynamic model is used (keyword =  $OPT\_DM\_2019$ ). This control is visible only when the single-point box ⑨ is unchecked.

**Note** The differences in behavior are usually small. The old equations (versions 2019.0 and older) are used when this box is checked, in order to support version-to-version validation tests.

- ⑪ Checkbox with an option to ignore steering effects of the rear wheels ( $OPT\_DM = 2$ ). This control is only visible when the 10-point Driver Model option is enabled (i.e., the single-point box ⑨ is unchecked). This option is provided for compatibility with legacy versions of the VS Math Models (prior to 2004), however this option is scheduled to be removed in a future release of the software and its use is therefore not recommended.

When this box is unchecked (the default behavior), the Closed Loop Driver Model with 10 preview points will consider the steer effects of axle 2 when calculating a handwheel angle. This includes steer due to the suspension kinematics, kinematical effects resulting from compliance, and road wheel steer via an axle 2 steering system.

When this box is checked, the legacy behavior (prior to 2004) is enabled, and the steer effects of axle 2 are not considered in the Closed Loop Driver Model's calculation of the handwheel angle. Steer effects of axle 2 will still affect the dynamics of the vehicle, however.

Note that this functionality only applies to lead units with two axles. For lead units with more than two axles (i.e., a TruckSim lead unit with three axles), the steering effects of all axles other than the first one are not included as part of the Closed Loop Driver Model's calculation of the handwheel angle.

- ⑫ Low-speed limit for speed sensitivity in the driver steering model (keyword = VLOW\_DRIVER). For speeds above this limit, the preview distance is proportional to speed. With the single-point box ⑨ unchecked, the internal dynamic vehicle model (a 2D dynamic model) is used that depends on speed. If the vehicle speed drops below this limit, then the preview is based this speed.

When speed is less than 0.1 m/s (0.36 km/h) and the single-point box ⑨ is unchecked, the steering wheel angle from the controller is frozen.

- ⑬ Maximum steering wheel angle for the driver model (keyword = A\_SW\_MAX\_DM). This specifies the maximum angle that the steering wheel can be turned in either direction from center.
- ⑭ Maximum steering wheel angle rate (keyword = AV\_SW\_MAX\_DM). This specifies the maximum angular rate at which the steering wheel can be turned in either direction by the driver model.
- ⑮ Checkbox to include driver lag. This control is visible only when the single-point box ⑨ is unchecked (Figure 22). When the driver lag box is checked, an adjacent yellow field is shown for driver lag (keyword = TLAG\_DM). Steering wheel angles generated by the driver model are delayed by this amount of time to simulate the neuromuscular delay in people. A realistic value is about 0.15 seconds. Larger values can be used to simulate impaired drivers.

As the lag increases, the driver-vehicle system tends to over-correct to the point of instability.

If you are attempting to follow a path closely, without trying to simulate driver response dynamics, a value of 0.0 is recommended. In these cases, simply un-check the box.

- ⑯ Checkbox to have the driver model steer by torque. When checked, three more fields are shown: ⑰, ⑱, and ⑲ (Figure 23). The DM still calculates a target steering wheel angle, but instead of applying it regardless of the implied torque, it instead uses a controller to generate steering wheel torque as needed to generate the angle requested by the controller.

The screenshot shows a software interface for driver model parameters. A checkbox labeled 'Driver model steers by torque' (⑯) is checked. Below it, a yellow field for 'Maximum control torque' (⑰) contains the value '50'. To the right, there are two more yellow fields: 'Controller stiffness' (⑱) with the value '25' and 'Controller damping' (⑲) with the value '0.5'. The units for these fields are 'N-m' and 'N-m/deg' respectively.

Figure 23. More parameters are used when the driver models steers by torque.

- ⑪ Maximum steering wheel torque allowed for DM to apply (keyword = M\_TRQ\_CON\_MAX\_DM).
- ⑫ Torsional stiffness used by DM to generate steering torque (keyword = K\_TRQ\_CON\_DM).
- ⑬ Torsional damping used by DM to generate steering torque (keyword = D\_TRQ\_CON\_DM).

## Visualization of the Target Path in VS Visualizer

The lower-right portion of the screen has controls to display a dashed line at the location of the target path in animations.

- ⑭ When the box is checked, additional controls are displayed to specify the appearance of the dashed line.

To show the line, rectangular shape objects are automatically generated for VS Visualizer according to the values set in these controls.

Three controls specify the size of the generated dash shapes: the width of the line in meters, centered on the target path, the length of each dash segment (the space between segments is the same as the segment length), and the resolution. Each segment is composed of one or more shapes, where each segment is broken into shapes of length = resolution. Smaller values make the displayed line segments follow hills and road elevation changes more closely, but may substantially increase the amount of data loaded by the VS Visualizer.

A parameter “dZ” specifies the height at which the dashes “float” above the 3D road surface. Positioning the shapes at the same height causes interference between textures, resulting in degraded image quality and flickering. Displacing them vertically by a small amount avoids this.

The color of the dashes is specified by entering RGB values in the color field or choosing a color from the palette control.

If the reference path for DM is not looped, then the shapes are generated to cover a range of station covered in the simulation. To determine this, the VS Browser reads the Echo files generated at the beginning and end of the run, scanning to find the station at the start and stop of the run. The keyword SSTART (a parameter used to initialize the vehicle) specifies station at the start of the run; the keyword SV\_STATION (a state variable) identifies the station at the end of the run.

If the reference path for the driver is looped, then shapes are generated to cover the entire loop.

<b>Note</b>	The path information used to generate the shapes is taken from the data at the start of the run. If the path is changed during the run with Events or VS Commands, those changes will not be shown. The option to visualize a path is mainly intended for simulations in which the same path is used for the entire run.
-------------	--

## Working with Multiple LTARG and Path Datasets

The Configurable Function LTARG is used by the driver model and can also be used to control moving objects that mimic traffic vehicles or other custom applications. The current number of active LTARG dataset is a system parameter with keyword N\_LTARG.

Before the VS Math Model reads any dataset Parsfiles, N\_LTARG has a value of zero. Each time a dataset from this library (**Control: Steering by the Closed-loop Driver Model**) is read, a new dataset is added if the ID is set automatically or if a new custom ID is specified. New LTARG datasets may also be added using the VS Command SET\_ILTARG\_FOR\_ID as described in the Help document **Help > Paths and Road Surfaces**.

Each dataset has an ID specified with the parameter LTARG\_ID; e.g., the ID for the first dataset is LTARG\_ID(1), for the second dataset it is LTARG\_ID(2), etc.

LTARG\_ID\_DM is the ID number of the LTARG dataset used by the driver model. It has a default value of 1, and can be set to any positive number up to N\_LTARG, or to any value that matches the LTARG\_ID of an existing dataset. If set to zero then the lateral offset from the path stays at zero. The LTARG\_ID\_DM value can be overridden by specifying a value after several LTARG datasets have been created to assign a specific target to the driver model. LTARG\_ID\_DM may also be changed during a VS Event. As described earlier (page 5), the custom ID option ② (Figure 19, page 26) is recommended if multiple LTARG datasets will be used in a simulation.

Each moving object has a parameter LTARG\_ID\_OBJ that links an LTARG dataset to the object if the object is also associated with a path via the parameter PATH\_ID\_OBJ (please see the Help document **Help > ADAS Sensors and Moving Objects** for more information).

The independent variable used in the LTARG function for DM is Station, the S coordinate of the vehicle sprung mass coordinate system origin relative to a path whose ID is specified with the parameter PATH\_ID\_DM. As noted in an earlier section (page 11), changing paths might require resetting Station to represent the closest point on the new path.

## The Built-in Closed-Loop Speed Controller (SC)

CarSim and TruckSim include a built-in closed-loop speed controller (SC) that will adjust throttle and braking as needed to respond to target speed or acceleration. If a minimal powertrain is installed (INSTALL\_POWERTRAIN = 0, Figure 24), SC will generate wheel torques using a simple model with parameters for the available power and the distribution of drive torque on the vehicle axles.

```
527 !-----
528 ! POWERTRAIN
529 !-----
530 INSTALL_POWERTRAIN 0 ! Powertrain type: 0 -> Simple, 1 -> FWD, 2 -> RWD, 3 -> AWD,
531 ! 7 -> AVL Cruise [L]
532
533 R_REAR_DRIVE_SC 0 ; - ! Ratio: [rear axle drive torque]/[total drive torque] [I]
534 PMAX_SC 125 ; kW ! Maximum power available for NO_PT (INSTALL_POWERTRAIN
535 ! NO_PT) [I]
536
```

Figure 24. Simple powertrain option for speed control.

The target speed for SC may be set from five library screens:



1. **Control: Speed (Closed Loop) Using Target Speed:** target speed and SC parameters for target speed mode.
2. **Control: Speed (Closed Loop) Using Path Preview:** speed limit and SC parameters for path preview mode.
3. **Run Control** (constant target speed or link to an SC dataset).
4. **Procedures** (constant target speed or link to an SC dataset).
5. **Events** (constant target speed, link to an SC dataset, or custom settings).

The first two screens are used to set most of the SC parameters along with a Configurable Function dataset for target speed (#1, **Target Speed**) or multiple Configurable Function datasets for limits in speed and acceleration (#2, **Path Preview**). The **Run Control**, **Procedures**, and **Events** screens support links to the first two libraries, but also provide direct access to SC if the target speed is constant and default SC parameter values are acceptable.

## Installing the Speed Controller

The speed controller is not initially installed; it must be installed with the command `INSTALL_SPEED_CONTROLLER`.

In most cases, SC is installed if the target speed is set from any of the five libraries listed in the previous subsection. However, in some scenarios, the simulation will start with the vehicle throttle set in open-loop mode, and later switch to SC by triggering a VS Event. SC cannot be installed after the simulation starts; instead, the VS Math Model will generate an error message and stop. (This is because SC installs differential equations, and the VS Math Model requires all differential equations to be defined before the simulation starts.)

For simulations that do not use SC at the start, but will use it later in the run, a checkbox on the Vehicle screen ① (Figure 25) may be used to ensure SC will always be available when simulating that vehicle. Another option is to put the command `INSTALL_SPEED_CONTROLLER` in a miscellaneous yellow field of the **Run Control** or **Procedure** dataset.

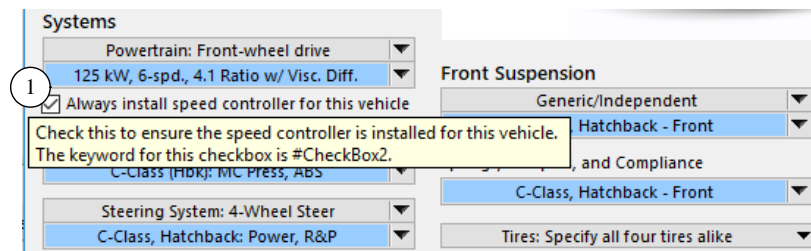


Figure 25. Checkbox on Vehicle screen to install SC with a vehicle.

Table 6 lists the output variables that are provided by SC. These variables are not available unless SC is installed. Note that import versions are available for the two command/target variables.



Table 6. SC output variables.

Short Name	Units	Full Name	Import
Ax_SCcmd	g	Ax command for speed controller	IMP_AX_SC
Ax_SCrq	g	Ax request from speed controller	
Pwr_SC	kW	Power from speed control (no PT)	
Pwr_SCrq	kW	Power requested from speed ctrl	
VxTarget	km/h	Target speed	IMP_SPEED
Vx_Err	km/h	Speed controller error	
Vx_IErr	m	Integrated speed control error	

**Note** If multiple vehicles are defined, the Import and output names have a suffix for vehicles 2 and higher indicating the vehicle number, e.g., Ax\_SCcmd\_2 for vehicle 2.

The SC parameters are listed together in the Echo file (Figure 26). Support is always provided for the SPEED\_TARGET Configurable Function, used by SC and moving objects. Up to 200 datasets can be defined for this function, with the number being shown with the keyword N\_SPEED\_TARGET (line 753). If installed by the command INSTALL\_SPEED\_CONTROLLER, the command is listed next (line 756). The comments and remaining parameters are shown only if SC is installed.

```

750 ! -----
751 ! DRIVER MODEL: SPEED CONTROLLER
752 ! -----
753 N_SPEED_TARGET      1 ! Number of SPEED_TARGET datasets for driver model, moving
754                       ! objects, and VS Commands that are written below: 0 - 200
755
756 INSTALL_SPEED_CONTROLLER ! VS Command to install the built-in speed controller
757 ! The speed controller uses throttle and braking controls to follow target speed
758 ! specified as a function of time and/or station along reference path PATH_ID_DM.
759 ! The target speed is specified with Configurable Function SPEED_TARGET. If the
760 ! speed is based on the path, then acceleration limits are specified using
761 ! functions SPEED_AX_BRAKE, SPEED_AX_THROTTLE, SPEED_AY_LEFT, and SPEED_AY_RIGHT.
762
763 OPT_SC                3 ! Speed controller: 0 -> Off (open-loop), 1 - 3, target speed
764                       ! is function of time and station, 4 -> target speed is
765                       ! determined using path preview, 5 -> use acceleration
766                       ! command Ax_SCcmd [I]

```

Figure 26. The top of the speed controller section of the Echo file.

The parameter OPT\_SC specifies the mode for SC (line 763). Based on the mode, other parameters are shown. Target speed can be specified as a predefined function of time and station, or it can be calculated dynamically by previewing the target path and considering the curvature and possibly the 3D properties of the road surface along the path (OPT\_SC = 4). Another option is that SC works with an acceleration command (OPT\_SC = 5).

## Interactions with Open-Loop Brake and Throttle Commands

Use of SC does not disable the open-loop braking control. If a nonzero open-loop braking command is specified, whether by a linked dataset using the built-in configurable function, VS Commands, use of braking import variables, or other means, any open-loop braking contributions are added to the command from the closed-loop speed control.

The same is true with throttle; if the vehicle has a full powertrain specified (rather than the internal minimum powertrain), then all contributions to open-loop throttle (built-in configurable function, VS Commands, import variables, etc.) are added to the throttle command from SC.

This means that open-loop control can be used together with closed-loop speed control to augment the performance of the closed-loop control logic to simulate, for example, systems that intervene to supplement or override control commanded by a human driver.

Be aware that CarSim and TruckSim powertrains require that throttle be limited to the range of 0.0 to 1.0. If the open-loop throttle plus the throttle from the speed controller plus possible imported throttle are greater than 1.0, the throttle is automatically set to 1.0. In the same way, if the total throttle is negative, it is automatically set to zero.

When switching between open-loop and closed-loop control with a VS Event, any open-loop control should be set to zero unless such augmentation is intended.

## Operation of SC

When it is in use, SC has two internal steps used to control the vehicle:

1. Request longitudinal acceleration ( $A_{x\_SCrq}$ ) based on target speed or acceleration and the current state of the vehicle motion.
2. Apply spin torque to the wheels of the lead unit using the powertrain and possibly brakes, to try to achieve  $A_{x\_SCrq}$ .

### *About speed, acceleration, and pitch*

Several variables are available in CarSim and TruckSim models to represent longitudinal speed and acceleration (Table 7).

The velocities and accelerations of the loaded sprung mass CG are based on the local body-fixed X-Y-Z axis directions and are used in the internal multibody equations of motion. The X axis is affected by pitch, which is in turn affected by the road grade (hills), the vehicle loading, and dynamic pitching. Neither  $V_{x\_SM}$  nor  $A_{x\_SM}$  are used within SC.

<b>Note</b>	The names shown in Table 7 are for the typical case in which the VS Math Model has a single vehicle. If multiple vehicles are defined, the output names have a suffix for vehicles 2 and higher indicating the vehicle number for $V_{xz\_Fwd}$ (e.g., $V_{xz\_Fwd\_2}$ ), and the unit number for all other variables in the table.
-------------	--

Table 7. Output variables for vehicle lead unit acceleration and velocity.

Name	Description	Speed controller
Ax	Components of acceleration of total unit mass CG in ISO/SAE directions	Not used because X direction is horizontal and not suitable for hills
Ay		
Az		
Ax_Rd	Components of acceleration of total unit mass CG in VS Road directions	Ax_Rd is used by SC when OPT_SC = 5
Ay_Rd		
Az_Rd		
Ax_SM	Components of acceleration of sprung mass CG in body X, Y, Z directions	Not used because X direction is affected by vehicle pitch
Ay_SM		
Az_SM		
Vx	Components of velocity of total unit mass CG in ISO/SAE directions	Not used because X direction is horizontal and not suitable for hills
Vy		
Vz		
Vx_Rd	Components of velocity of total unit mass CG in VS Road directions	Not used due to time lag in versions < 2018.0
Vy_Rd		
Vz_Rd		
Vx_SM	Components of velocity of sprung mass CG in body X, Y, Z directions	Not used because X direction is affected by vehicle pitch
Vy_SM		
Vz_SM		
Vxz_Fwd	$\sqrt{Vx\_SM^2 + Vz\_SM^2}$ , with same sign as Vx_SM	Vxz_Fwd is used by SC when OPT_SC = 1, 2, 3, 4

The ISO/SAE axis directions (start with global X-Y-Z axes, then rotate about the Z axis by vehicle yaw) are used to obtain X-Y-Z components of the acceleration and velocity of the total vehicle mass, including instant positions of wheels and other parts of the unsprung masses. Because they do not take road grade into account, neither Vx nor Ax are used within SC.

The Vxz\_Fwd speed was introduced years ago specifically for use in SC. Although the Z component includes dynamic ride motions, these are outside the frequency response of SC and do not degrade the performance. On the other hand, the steady component of Vz\_SM due to quasi-static pitch related to the ground slope or vehicle loading ensures that the Vxz\_Fwd speed matches the speed of the sprung mass CG parallel to the road.

The VS Road axes are oriented such that the X and Y axes are parallel with the road surface at a point matching the vehicle aerodynamic reference point. The Ax\_Rd acceleration is used in SC when operated in acceleration control mode (OPT\_SC = 5).

**Note** Until version 2018.0, internal timing of when variables are calculated made Vxz\_Fwd a better choice for use in SC than Vx\_Rd. The difference in behavior is very minor.

### Obtaining the requested acceleration based on target speed

With the first three options (OPT\_SC = 1, 2, 3), the SPEED\_TARGET Configurable Function is used to determine a target speed, and SC uses feedback to control the powertrain and braking. If the path-preview option is used (OPT\_SC = 4), the SPEED\_TARGET function is used to calculate a maximum speed limit that might be reduced due to acceleration limits.

Figure 27 shows the SC parameters that are used for an example double-lane change maneuver in which the speed is intended to be constant.

```

799
800 OPT_SC          3 ! Speed controller: 0 -> Off (open-loop), 1 - 3, target speed
801                  ! is function of time and station, 4 -> target speed is
802                  ! determined using path preview, 5 -> use acceleration
803                  ! command Ax_SCcmd [I]
804 OPT_AUTO_RESET_IC 1 ! [D] Reset integral of speed error Vx_Err when Vx_Err changes
805                  ! sign AND |Vx_Ierr| > VX_IERR_DEAD_SC: 1 -> yes, 0 -> no
806 OPT_BK_SC        0 ! [D] Braking and SC: 0 -> turn controller off while brakes
807                  ! are applied, 1 -> controller can use the brakes
808 OPT_SC_ENGINE_BRAKING 0 ! Use engine braking to control speed? 0 -> no, 1 -> yes
809 SPEED_ID_SC      1 ! SPEED_TARGET_ID for speed controller [I]
810 SPEED_KP         0.14 ; s/m ! [D] Speed controller: proportional control gain
811 SPEED_KP3        0 ; s3/m3 ! [D] Speed controller: cubic (verr^3) control gain
812 SPEED_KI         0.16 ; 1/m ! [D] Speed controller: integral control gain
813 VX_IERR_DEAD_SC  1 ; m ! [D] Integral control deadband for auto reset
814

```

Figure 27. Speed controller parameters used for double-lane change example.

When the SPEED\_TARGET function is used (OPT\_SC = 1, 2, 3, or 4), SC uses the dataset whose SPEED\_TARGET\_ID matches the parameter SPEED\_ID\_SC (line 809).

**Note** In much older versions, OPT\_SC options 1, 2, and 3 were used to distinguish between a constant speed, speed as a function of time, and speed as a function of station. Given that the same Configurable Function SPEED\_TARGET is now used for all three cases, the values 1, 2, and 3 all have the same effect.

Given a target speed  $V_{\text{target}}$ , SC calculates the requested acceleration with the equation:

$$A_{\text{xSCRq}} = K_p V_{\text{err}} + K_i V_{\text{Ierr}} + K_{p3} V_{\text{err}}^3 \quad (14)$$

where

$$V_{\text{err}} = V_{\text{target}} - V_{\text{xz\_fwd}} \quad V_{\text{Ierr}} = \int V_{\text{err}} dt \quad (15)$$

$$V_{\text{xz\_fwd}} = \text{sign}(1, V_{\text{x\_SM}}) \sqrt{V_{\text{x\_SM}}^2 + V_{\text{z\_SM}}^2} \quad (16)$$

Speeds are expressed internally in m/s and the coefficients  $K_p$ ,  $K_i$ , and  $K_{p3}$  are defined such that the requested acceleration  $A_{\text{xSCRq}}$  is dimensionless (g). These coefficients appear in the Echo file (Figure 27) with keywords SPEED\_KP, SPEED\_KI, and SPEED\_KP3, respectively.

<b>Note</b>	As indicated in equation 14, $A_{xSCrq}$ (the requested acceleration) is calculated based on specified coefficients and the speed and integral errors. If the target speed differs substantially from the current speed, the requested acceleration may be very large in magnitude. The actual acceleration is limited by the physics of the vehicle powertrain, brake system, and tire/road friction.
-------------	--

### *About integral control*

Integral control forces the speed error to zero when the target speed has a constant or slowly changing value. The integral of longitudinal speed is longitudinal distance, so if the distance error  $Vx\_IErr$  accumulates, the speed is adjusted accordingly.

Be aware that integral control also adds dynamics to the closed-loop system that can cause overshoot and instabilities if the target speed goes through large changes. To help remedy this, the controller always resets the integral  $Vx\_IErr$  to zero if the speed  $Vxz\_Fwd$  has a different sign than the target speed  $VxTarget$ , which occurs when the vehicle stops or spins out. Further, the controller has two parameters that handle other cases for resetting  $Vx\_IErr$ :  $OPT\_AUTO\_RESET\_IC$  (line 767, Figure 27) and  $VX\_IERR\_DEAD\_SC$  (line 776). These parameters are shown in the Echo file only when  $SPEED\_KI$  is nonzero.

When  $OPT\_AUTO\_RESET\_IC = 1$ , the controller will automatically reset  $Vx\_IErr$  to zero when  $Vx\_Err$  changes sign and the magnitude of  $Vx\_IErr$  is greater than the “dead zone” parameter  $VX\_IERR\_DEAD\_SC$ .

The default values ( $OPT\_AUTO\_RESET\_IC = 1$  and  $VX\_IERR\_DEAD\_SC = 1m$ ) work well for most examples. They are not linked to GUI controls, but advanced users can change them by means of miscellaneous yellow fields.

### *Obtaining target speed*

When  $OPT\_SC = 1, 2$ , or  $3$ , the target speed is provided by the  $SPEED\_TARGET$  Configurable Function as a constant or a function of time, station, or both, using the dataset whose  $SPEED\_TARGET\_ID$  value matches  $SPEED\_ID\_SC$ .

When  $OPT\_SC = 4$ , the  $SPEED\_TARGET$  Configurable Function is used to define a maximum speed limit. The target speed is obtained by previewing the path for DM ahead of the vehicle. (This option is only available if DM is active:  $OPT\_DM > 0$ .) More parameters are used in this mode (Figure 28).

The comments at the top of this section of the Echo file mention the Configurable Functions associated with speed control; along with  $SPEED\_TARGET$ , there are the functions  $SPEED\_AX\_BRAKE$ ,  $SPEED\_AX\_THROTTLE$ ,  $SPEED\_AY\_LEFT$ , and  $SPEED\_AY\_RIGHT$ . Datasets for those functions are listed in the Echo file in alphabetical order with other Configurable Functions.

```

849 !-----
850 ! DRIVER MODEL: SPEED CONTROLLER
851 !-----
852 N_SPEED_TARGET      4 ! Number of SPEED_TARGET datasets for driver model, moving
853                       ! objects, and VS Commands that are written below: 0 - 200
854
855 INSTALL_SPEED_CONTROLLER ! VS Command to install the built-in speed controller
856 ! The speed controller uses throttle and braking controls to follow target speed
857 ! specified as a function of time and/or station along reference path PATH_ID_DM.
858 ! The target speed is specified with Configurable Function SPEED_TARGET. If the
859 ! speed is based on the path, then acceleration limits are specified using
860 ! functions SPEED_AX_BRAKE, SPEED_AX_THROTTLE, SPEED_AY_LEFT, and SPEED_AY_RIGHT.
861
862 OPT_SC              4 ! Speed controller: 0 -> Off (open-loop), 1 - 3, target speed
863                       ! is function of time and station, 4 -> target speed is
864                       ! determined using path preview, 5 -> use acceleration
865                       ! command Ax_SCcmd [I]
866 OPT_AUTO_RESET_IC   1 ! Reset integral of speed error Vx_Err when Vx_Err changes
867                       ! sign AND |Vx_IErr| > VX_IERR_DEAD_SC: 1 -> yes, 0 -> no [I]
868 OPT_SC_3D           0 ! Path preview for SC: 1 -> account for 3D ground curvature, 0
869                       ! -> assume flat level ground [I]
870 OPT_SC_SKILL         2 ! Speed controller skill: 2 -> high, 1 -> medium, 0 -> low [I]
871 OPT_SC_ENGINE_BRAKING 1 ! Use engine braking to control speed? 0 -> no, 1 -> yes [I]
872 OPT_SC_2018         0 ! [D] Option: 0 -> use g/MPa units for BK_PERF_SC, 1 ->
873                       ! multiply BK_PERF_SC by G for old datasets (2018.0 and
874                       ! older) [I]
875 SPEED_ID_SC          2000 ! SPEED_TARGET_ID for speed controller [I]
876 BK_PERF_SC           0.15 ; g/MPa ! Approximate vehicle decel per unit pressure [I]
877 PBK_CON_MAX_SC       10 ; MPa ! Maximum master cylinder pressure used by SC [I]
878 SPEED_CURV_LENGTH    4 ; m ! Speed controller: length of path used to calculate
879                       ! curvature with 3 points (both ends and the mid-point) [I]
880 SPEED_KP             0.3 ; s/m ! Speed controller: proportional control gain
881 SPEED_KP3            0 ; s3/m3 ! Speed controller: cubic (verr^3) control gain
882 SPEED_KI             0.5 ; 1/m ! Speed controller: integral control gain
883 SPEED_PREVIEW        100 ; m ! Speed controller: path preview distance [I]
884 SPEED_PREVIEW_START  0 ; m ! Speed controller: path preview start (distance in front
885                       ! of sprung mass origin) [I]
886 SPEED_PREVIEW_STEP    1 ; m ! Speed controller: preview interval (resolution) [I]
887 VX_IERR_DEAD_SC      2 ; m ! Integral control deadband for auto reset [I]
888

```

Figure 28. Parameters created by the `INSTALL_SPEED_CONTROLLER` command.

Using the target path, SC calculates target speed based on the following considerations:

1. The target speed is limited to a speed limit defined with the `SPEED_TARGET` function with the dataset identified by the `SPEED_ID_SC` parameter.
2. The target speed is possibly reduced in turns in order to keep lateral acceleration  $A_y$  within limits specified separately for left- and right-hand turns using Configurable Functions `SPEED_AY_LEFT` and `SPEED_AY_RIGHT`. These can be functions of both station and speed.
3. The target speed is possibly reduced when approaching lowered speeds to account for realistic braking, to keep combined longitudinal and lateral acceleration ( $A_x$  and  $A_y$ , respectively) within specified limits. The deceleration limit is obtained from the Configurable Function `SPEED_AX_BRAKE`, a function of station and speed.
4. The target speed is possibly reduced during throttle, to keep combined longitudinal and lateral acceleration within specified limits. The forward acceleration limit is obtained from the Configurable Function `SPEED_AX_THROTTLE`, a function of station and speed.
5. The acceptable acceleration levels may be adjusted by considering the slope and vertical of the road surface, as enabled with the parameter `OPT_SC_3D`.

6. The acceptable combination of lateral and longitudinal acceleration is determined one of three ways, based on the parameter OPT\_SC\_SKILL, which may be given a value of 0, 1, or 2 (Figure 29).

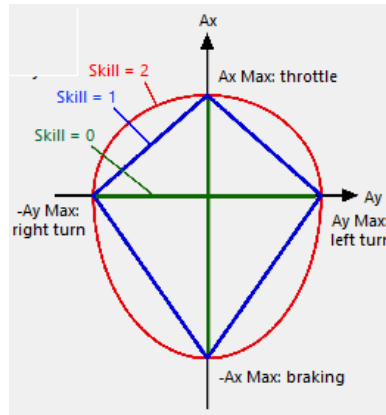


Figure 29. Three skill levels used to combine lateral and longitudinal acceleration.

### Acceleration control mode

SC has a direct acceleration control mode (OPT\_SC = 5) where the above equations for generating  $A_{x\_SCrq}$  are not used. Instead, a command variable  $A_{x\_SCcmd}$  may be set directly by VS Commands or by import (IMP\_AX\_SC). With this option, there is no target speed, and therefore no parameters involving specifying target speed, or using speed error. Figure 30 shows how the SC section of the Echo file appears when OPT\_SC = 5. Many of the parameters seen earlier (Figure 28) are gone, but one new parameter is shown: ACCEL\_KP\_SC (line 787).

```

769 !-----
770 ! DRIVER MODEL: SPEED CONTROLLER
771 !-----
772 N_SPEED_TARGET      0 ! [D] Number of SPEED_TARGET datasets for driver model, moving
773                       ! objects, and VS Commands that are written below: 0 - 200
774
775 INSTALL_SPEED_CONTROLLER ! VS Command to install the built-in speed controller
776 ! The speed controller uses throttle and braking controls to follow target speed
777 ! specified as a function of time and/or station along reference path PATH_ID_DM.
778 ! The target speed is specified with Configurable Function SPEED_TARGET. If the
779 ! speed is based on the path, then acceleration limits are specified using
780 ! functions SPEED_AX_BRAKE, SPEED_AX_THROTTLE, SPEED_AY_LEFT, and SPEED_AY_RIGHT.
781
782 OPT_SC                5 ! Speed controller: 0 -> Off (open-loop), 1 - 3, target speed
783                       ! is function of time and station, 4 -> target speed is
784                       ! determined using path preview, 5 -> use acceleration
785                       ! command Ax_SCcmd [I]
786 OPT_SC_ENGINE_BRAKING 1 ! Use engine braking to control speed? 0 -> no, 1 -> yes
787 ACCEL_KP_SC           5 ; - ! Control gain applied to accel error: Ax_SCcmd - Ax_Rd
788 BK_PERF_SC            0.09 ; g/MPa ! Approximate vehicle decel per unit pressure
789 PBK_CON_MAX_SC        20 ; MPa ! Maximum master cylinder pressure used by SC
790

```

Figure 30. Echo file when acceleration control is selected.

In this mode, the SC equation to obtain  $A_{x\_SCrq}$  ( $A_{xSCrq}$ ) is simply:

$$A_{xSCrq} = A_{xRd} + K_{Acc} (A_{xSCcmd} - A_{xRd}) \quad (17)$$

where  $K_{Acc}$  is the parameter ACCEL\_KP\_SC,  $A_{xSCcmd}$  is the command  $A_{x\_SCcmd}$ , and  $A_{xRd}$  is the vehicle longitudinal acceleration parallel to the road surface,  $A_{x\_Rd}$ .



Figure 31 shows a plot with three accelerations for an example in which longitudinal acceleration was specified twice to brake the vehicle to a stop in a 3-point turn. The command acceleration (the red line in the top plot) is zero when  $OPT\_SC \neq 5$ . The plot shows it has non-zero values at around 12s, when it has a negative value, and at 25s, when it has a positive value. In both cases, the acceleration control was used to brake the vehicle to a stop. In the first instance, the vehicle was going forward, and the acceleration command was negative. In the second instance, the vehicle was driving in reverse, and the acceleration command was positive.



Figure 31. Plot of acceleration variables with  $OPT\_SC = 5$ .

Note how the acceleration request (green line) tries to bring the vehicle acceleration (blue line) closer to the command (red line).

### Obtaining throttle and braking based on requested acceleration

Given a requested acceleration  $Ax\_SCrq$ , SC generates braking and powertrain requests.

If a powertrain is installed ( $INSTALL\_POWERTRAIN > 0$ ), then SC will request power using the throttle. As noted earlier (see Figure 24, page 31), a simplified model is used when  $INSTALL\_POWERTRAIN = 0$ , involving just an available power parameter  $P_{MAX\_SC}$  and a ratio of drive provided for the rear axle ( $R\_REAR\_DRIVE\_SC$  for CarSim) or each axle ( $R\_DRIVE\_SC(1)$ ,  $R\_DRIVE\_SC(2)$ , ... for TruckSim). In this case, the drive torque is calculated for each axle and applied directly to the wheels on that axle.

If a powertrain is installed, the maximum power available is required by the speed controller. For internal powertrains, this information is derived from the parameters specified in the powertrain datasets. However, if an external powertrain is used, this information must be supplied by the user via the import variable  $IMP\_PWR\_ENGINE\_AV$ .  $IMP\_PWR\_ENGINE\_AV$  should be used for all external powertrain configurations (i.e., internal combustion engine, electric motors, hybrid



systems, etc.). When the import variable was created, an internal combustion engine was the only available powertrain type. The variable name reflects this history.

If a full powertrain is installed, then engine braking is used if `OPT_SC_ENGINE BRAKING = 1`.

When `OPT_SC = 1, 2, or 3`, the use of the brake system is optional, and is specified with the `OPT_BK_SC` parameter. If `OPT_BK_SC = 0`, SC acts like a typical cruise control option; SC is disabled when the brakes are applied. If `OPT_BK_SC = 1`, SC is allowed to apply brakes. SC always allows the use of brakes in the path-preview or acceleration control mode (`OPT_SC = 4 or 5`).

Several SC parameters are used to determine the braking control (Figure 32). The parameter `BK_PERF_SC` is a system-level coefficient that relates vehicle deceleration to master cylinder brake control pressure. Given that SC is a closed-loop controller, this coefficient does not need to be perfect — it is only an approximation. If the brake system is setup to use pedal force as the main input, then a second coefficient is needed to relate pedal force to master cylinder pressure: `FPD_PERF_SC`. The maximum control pressure available to SC is specified with the parameter `PBK_CON_MAX_SC`.

814	<code>OPT_SC_ENGINE BRAKING</code>	1	! Use engine braking to control speed? 0 -> no, 1 -> yes [I]
815	<code>OPT_SC_2018</code>	0	! [D] Option: 0 -> use g/MPa units for <code>BK_PERF_SC</code> , 1 ->
816			! multiply <code>BK_PERF_SC</code> by G for old datasets (2018.0 and
817			! older)
818	<code>SPEED_ID_SC</code>	1	! <code>SPEED_TARGET_ID</code> for speed controller [I]
819	<code>BK_PERF_SC</code>	2	; g/MPa ! Approximate vehicle decel per unit pressure [I]
820	<code>FPD_PERF_SC</code>	0.0036	; MPa/N ! Approximate ratio: Brake M/C pressure per N pedal
821			! force
822	<code>PBK_CON_MAX_SC</code>	15	; MPa ! Maximum master cylinder pressure used by SC [I]

Figure 32. SC parameters related to the use of braking.

When preparing version 2018.1, a mistake was found involving the internal usage of the parameter `BK_PERF_SC`. In versions 2018.0 and older, the coefficient was incorrectly multiplied by G (9.80665 m/s<sup>2</sup>). Given that this coefficient is an approximation, it turns out that many example simulations perform acceptably either way. However, others require the coefficient to represent the system behavior more closely. In these cases, the preferred solution when using an old dataset is to multiply the value by G to replicate the old behavior. Another option is provided by the parameter `OPT_SC_2018`, which will perform the multiplication internally. This may be helpful when performing version-to-version validation studies.

### Driving in reverse

The speed controller was originally designed under the assumption that the vehicle is travelling forward. Starting with version 2019.1, DM supports driving in reverse. SC also supports driving in reverse, with some restrictions:

1. Driving in reverse is not supported using the path preview option (`OPT_SC = 4`).
2. When using the basic speed controller with a target speed (`OPT_SC = 1, 2, or 3`):
  - a. The powertrain must be set to use an appropriate gearing (forward or reverse) matched to the target speed, and

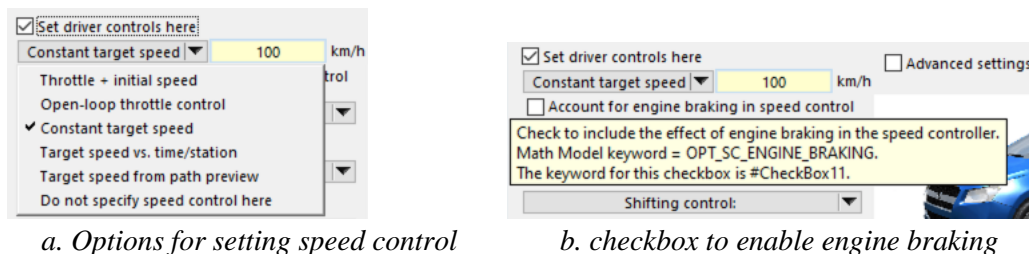
- b. the `SPEED_TARGET` Configurable Function datasets should not cause the speed target to change sign. This is, it must always be positive (for conventional forward driving) or always negative (for driving in reverse).
- c. Changing between forward and reverse must be done with a VS Event, which changes both the target speed dataset and the powertrain gearing direction.

## Setting Speed from Run Control, Procedures, and Events Screens

In simulations where the vehicle speed is intended to be constant, SC may be enabled directly from the **Run Control** screen, the **Procedures** screen, and the **Events** screen.

### Setting target speed from the Run Control screen

The **Run Control** screen has checkboxes for showing additional controls. If the **Show more options** box is checked, more checkboxes appear, including one labelled **Set driver controls here**. When this box is also checked, a drop-down control is shown for choosing among options to control speed (Figure 33a). Two of the options will show a link to the libraries described in following sections (**Target speed vs. time/station** and **Target speed from path preview**). There is also an option for **Constant target speed**, which causes the Browser to show a yellow field for the speed and a checkbox **Account for engine braking in speed control** (Figure 33b).



a. Options for setting speed control

b. checkbox to enable engine braking

Figure 33. Specifying SC and setting target speed from the Run Control screen.

This option activates SC, adds a new `SPEED_TARGET` dataset that specifies a constant whose value is provided in the yellow field, and enables or disables the engine braking option with the checkbox. More specifically, Figure 34 shows how the settings shown in Figure 33b is written in the Parsfile for the **Run Control** dataset to configure SC:

```

52
53 INSTALL_SPEED_CONTROLLER
54 Opt_SC 3
55 SET_ISPEED_FOR_ID 0
56 SPEED_ID_SC = SPEED_TARGET_ID
57 SPEED_TARGET_COMBINE ADD
58 SPEED_TARGET_S_CONSTANT 0
59 set_description SPEED_TARGET_ID Set from the Run Control screen
60 SPEED_TARGET_CONSTANT 100
61 OPT_SC_ENGINE_BRAKING 0
62 *SPEED 100
63

```

Figure 34. Portion of the Parsfile setting speed control for settings shown in Figure 33b.

- SC is installed (line 53).
- OPT\_SC is set to 3 (use target speed, line 54).
- The VS Command SET\_ISPEED\_FOR\_ID is used to add a new TARGET\_SPEED dataset and make the new dataset the default by setting ISPEED. The VS Command also sets the ID of the new dataset SPEED\_TARGET\_ID (line 55).
- The new ID is assigned to SPEED\_ID\_SC (line 56).
- The 2D Configurable function is given a station-dependent component that is a constant (SPEED\_TARGET\_S\_CONSTANT) of zero (line 57) and a time-dependent component that is also a constant (SPEED\_TARGET\_CONSTANT) of 100 km/h (line 60).
- The two components are added (line 57).
- Engine braking is disabled (OPT\_SC\_ENGINE BRAKING 0, line 61).

All other SC parameters (SPEED\_KP, SPEED\_KI, etc.) retain their existing values.

### *Setting target speed from the Procedures screen*

The **Procedures** screen has the same set of controls as the **Run Control** screen, located in the upper-left corner of the window. The same list of options is provided, and the commands written to the Parsfile are the same if the choice is made to use a constant target speed.

The Parsfile from a **Procedures** dataset linked from the **Run Control** screen will be read by the VS Math Model before the speed setting from the **Run Control** screen. Therefore, if different target speeds are set from the two screens, the settings from the **Run Control** screen (read last) will be used by SC.

### *Setting target speed from the Events screen*

The **Events** screen has several options for controlling speed. As a minimum, a drop-down control in the upper-left part of the window (1) (Figure 35a) is available for choosing options (Figure 35b). The options on the drop-down control are the same as those offered on the **Run Control** and **Procedures** screens. However, when the option **Constant target speed** is chosen, there is also an option to **Set a custom ID** (Figure 35a), as is done on the **Speed Control (Closed Loop) Using Path Preview** screen, described later (page 48). As with the other screens, a checkbox is available to **Account for engine braking in speed control** (3).

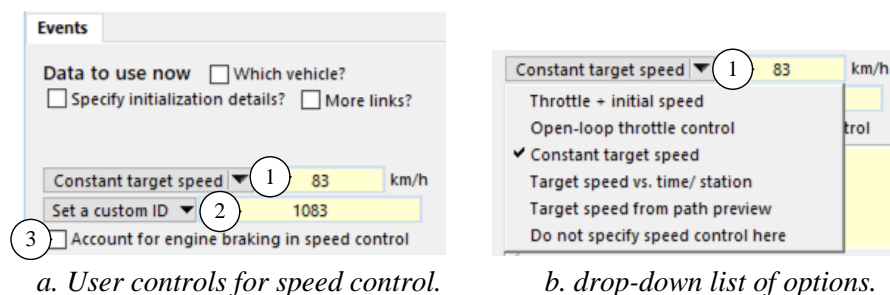


Figure 35. Options on the Events screen for controlling speed.

The option to use a custom ID is recommended if the **Events** dataset is expected to be used more than once, as described in following subsections. More information about this option is presented in a later subsection (page 56).

## Speed (Closed Loop) Using Target Speed: GUI Screen

Figure 36 shows the screen used to specify a target speed as a function of time and/or station. The top part of the screen shows a dataset for the SPEED\_TARGET Configurable Function. The bottom part shows properties of the closed-loop speed controller used for the vehicle.

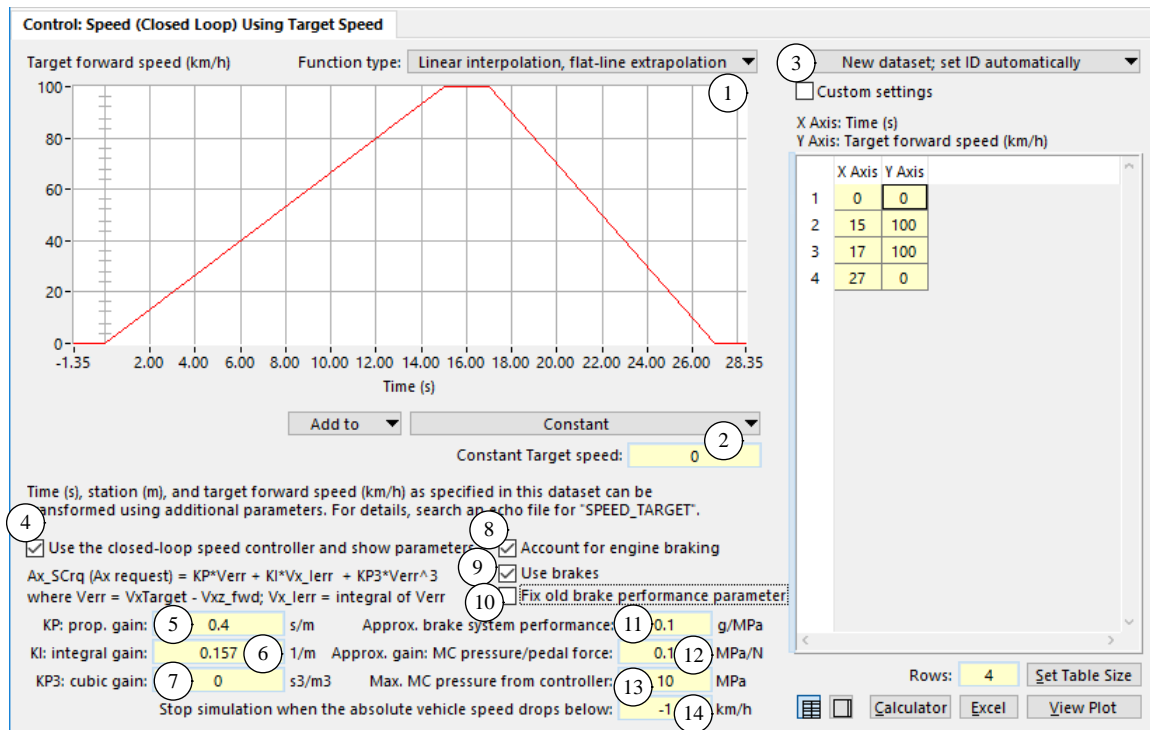


Figure 36. Speed target and closed-loop speed controller parameters.

This is a dual-purpose library, in the same manner as the Closed-Loop Steering library described earlier. It is used to define SPEED\_TARGET datasets that are used by moving objects, or for advanced simulations involving Events. In these cases, settings are not needed for SC.

The second purpose is to set most of the SC parameters.

### Target speed

This screen has the typical controls for setting up a Configurable Function that involves two independent variables. In the case of the function SPEED\_TARGET, the calculated variable is the target speed, the primary independent variable is time, and the secondary independent variable is station. The SPEED\_TARGET function is used by SC and for moving objects that mimic traffic vehicles, pedestrians, etc. VS Math Models support up to 200 SPEED\_TARGET datasets. These include user ID numbers that can help manage the use of multiple datasets.

- ① Drop-down list of function types for target speed. As with most Configurable Functions, this can be a constant, a coefficient, a nonlinear table specified with a set of numbers and a choice of interpolation and extrapolation methods, or a symbolic equation as described in the *VehicleSim Browser Reference Manual* and *VS Math Models Reference Manual*. If the calculation method is set to anything except a 2D interpolation method or a symbolic equation, then additional controls are shown for defining a possible sensitivity to station ②.
- ② Controls for specifying the sensitivity of the target speed to station. If the function type specified with the top control ① is not a 2D function or an equation, then additional controls are shown for specifying the effect of station on target speed with a secondary function. These options include a constant, a coefficient, and several tabular interpolation methods. The contribution from the secondary function (due to station) can either be added to the contribution of the primary function (due to time) or multiplied.
- ③ Drop-down control for setting the user ID. This Configurable Function includes a user ID (keyword = `SPEED_TARGET_ID`). The ID can be set automatically, or an ID can be specified that is 999 or higher. Details on setting and using ID numbers for Configurable Functions are described in the *VehicleSim Browser Reference Manual*.

If the ID is set automatically, the Parsfile for the dataset increments the number of `SPEED_TARGET` datasets (`N_SPEED_TARGET`) and assigns the new value of `N_SPEED_TARGET` to the ID, `SPEED_TARGET_ID`. For example, if there were already three `SPEED_TARGET` datasets, the automatically set ID for a new dataset will be `SPEED_TARGET_ID = 4`.

As described earlier (page 5), the custom ID option is recommended if multiple `SPEED_TARGET` datasets will be used in a simulation.

## SC Settings

The major SC parameters are set with the controls on the lower part of this screen.

- ④ Checkbox **Use the closed-loop speed controller and show parameters**. If this box is not checked, then all the controls ⑤ and higher are not shown (everything in the lower portion of the screen). Uncheck this box if the intent is to create a `SPEED_TARGET` dataset without affecting the built-in speed controller. For example, the `SPEED_TARGET` dataset may be used for Events or for controlling the speeds of moving objects.

When checked, the command `INSTALL_SPEED_CONTROLLER` is written in the Parsfile to ensure that the following parameters have been installed. Also, the SC parameter `SPEED_ID_SC` is automatically set to the ID of this dataset, `SPEED_TARGET_ID`.

If not checked, the `INSTALL_SPEED_CONTROLLER` command is not written, and there is no reference to any of the SC parameters.

- ⑤ Proportional gain coefficient  $K_p$  (keyword = `SPEED_KP`). Typical values are about 0.5. Setting the value higher causes the drive torque to be modulated more rapidly. Although this gain controls the rate of the control response, it does not force the speed error to zero. To make the vehicle speed match the target speed, integral control is needed ⑥.

- ⑥ Integral gain coefficient  $K_i$  (keyword = `SPEED_KI`). Integral control forces the speed error to zero when the target speed has a constant or slowly changing value. The integral of longitudinal speed is longitudinal distance, so if the distance error `Vx_IErr` accumulates, the speed is adjusted accordingly. When the target speed is constant, a  $K_i$  value of about  $0.5 \text{ m}^{-1}$  is typical. More information was presented in an earlier subsection (page 37).
- ⑦ Cubic gain coefficient  $K_{p3}$  (keyword = `SPEED_KP3`). Nonlinear cubic control generates an aggressive response when there are large differences between the target speed and vehicle speed. When the vehicle speed approaches the target speed, the cubic effect becomes negligible. This coefficient can be given a value of zero for constant or slowly changing speeds. For more dynamic speed changes, values of about 1.0 are typical.
- ⑧ Option to account for engine braking in the speed control (keyword = `OPT_SC_ENGINE_BRAKING`). When the simulated vehicle includes a full powertrain, the throttle is estimated using engine properties at the current engine speed (RPM). When this box is checked, the engine table is also checked for the current speed and zero throttle in order to provide better transitions between low throttle and braking.
- ⑨ Option to include braking in the speed control (keyword = `OPT_BK_SC`). When this box is checked, SC is allowed to use the brakes to slow the vehicle.

When this box is not checked, two things happen: (1) Applying the brakes disables the speed control, mimicking the typical behavior of a cruise control. (VS Events can be used to turn the speed control on again, if desired.); and (2) The speed controller uses the same control for deceleration as is used for acceleration (throttle or direct application of torque to the drive wheels, depending on whether the vehicle has a full or “minimal” powertrain).

- ⑩ Option to scale the brake performance parameter ⑪ in case it was set in an older version (2018.0 or earlier). The controller in old versions had a scaling error in which the parameter was inadvertently multiplied by G (9.80665). If this box is checked, the old scaling is applied.
- ⑪ Approximate brake system performance (keyword = `BK_PERF_SC`), used in the closed-loop controller. This value relates deceleration to brake pressure from the master cylinder. The feedback method used by the controller compensates for errors in this parameter, so an approximate estimate is usually sufficient. (It can be determined by performing a stopping test with open-loop brake control.) This field is visible only when box ⑨ is checked.
- ⑫ Brake pedal force to master cylinder pressure gain (keyword = `FPD_PERF_SC`). This value needs only to be a reasonable estimate and can be determined by performing a stopping test with open-loop brake control. This field is visible only when box ⑨ is checked. This parameter is used only when Pedal Force is selected on the Brake System screen.
- ⑬ Maximum master cylinder pressure that is allowed for the closed-loop controller (keyword = `PBK_CON_MAX_SC`).
- ⑭ Stop speed (keyword = `VLOW_STOP`). A simulation continues until one of several conditions occurs. One condition is when the absolute vehicle speed drops below this threshold.



The minimum speed must be given a value that is smaller than the initial vehicle speed or the target speed. To prevent the simulation from stopping due to low speed, set the threshold to a negative number (e.g., -1).

**Note** VLOW\_STOP is a model parameter shown in the Echo file in a section near the top with the title Vehicle Initialization, Limits, and Gravity. This parameter can be set from several screens involving speed control or braking.

### Speed (Closed Loop) Using Path Preview: GUI Screen

The speed controller can limit target speed based on of curvature in the reference target path, combined with driver aggressiveness, skill parameters, and possibly 3D surface properties. This is done with the **Control: Speed (Closed Loop) Using Path Preview** screen (Figure 37).

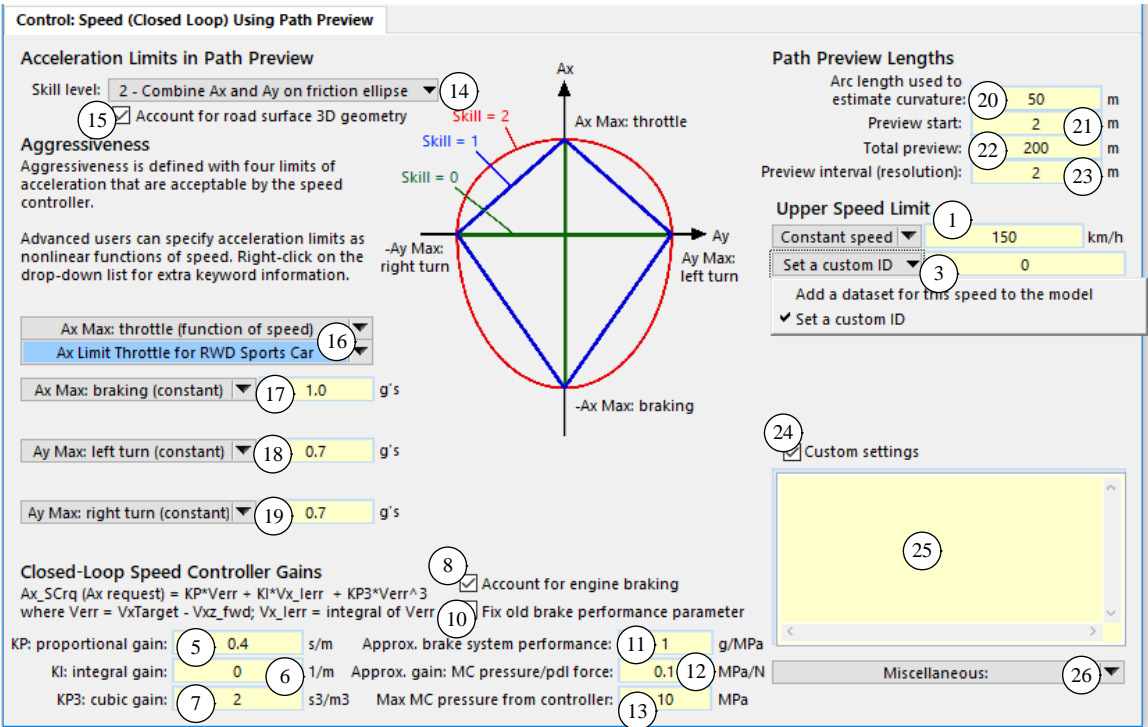


Figure 37. Speed Control Using Path Preview.

When you use a dataset from this library, the `INSTALL_SPEED_CONTROLLER` command is automatically applied to install SC, which is set to run in path preview mode (`OPT_SC = 4`). In this case, SC might reduce target speed by previewing the target path ahead of the vehicle, as described earlier (page 37).

The user controls on the bottom of this screen are the same as those shown on the other screen and described in the previous subsection (items (5) - (13)) in both Figure 36 and Figure 37).

Brake control is fundamental for the path preview mode and cannot be disabled; hence, the checkbox for disabling the brakes ((9) in Figure 36) does not exist on this screen.



The option to stop the run when a minimal speed is reached is also not included when operating with the path preview option; therefore, the minimal speed field is not included ((14) in Figure 36).

### *Upper speed limit*

- (1) The maximum target speed for SC is determined using the `SPEED_TARGET` Configurable Function. A dataset can be created here using a constant, or a link can be made to a dataset from the library screen described in the preceding section. Use the drop-down control to choose between these options.

If a constant is chosen, a yellow field is used to specify the value as shown in Figure 37. An ID is also needed to identify which `SPEED_TARGET` dataset will be used. Two options are available using the drop-down control (3).

If a function is chosen, a link is available to select a dataset from the library **Speed (Closed Loop) Using Target Speed** (Figure 36, page 44). The intent is that the linked dataset will define an upper limit target as a function of station. Internally, the speed controller uses the current path station  $S$  and  $T=0$  when using the `SPEED_TARGET` Configurable Function. The closed-loop controller gains ((5) - (13)) are read from the linked `SPEED_TARGET` dataset but will be overridden with the values read from this screen.

As noted earlier, the speed limit should not be negative; this mode for the speed controller does not work for driving in reverse.

- (3) Drop-down control for setting the user ID. This Configurable Function includes a user ID (keyword = `SPEED_TARGET_ID`). The ID can be set automatically, or an ID can be specified that is 999 or higher. Details on setting and using ID numbers for Configurable Functions are provided in the. *VehicleSim Browser Reference Manual*.

As described in the previous section (page 44), the custom ID option is recommended if multiple `SPEED_TARGET` datasets will be used in a simulation.

Regardless of whether the upper speed limit is set as a constant or by linking to a **Speed (Closed Loop) Using Target Speed** dataset, the SC parameter `SPEED_ID_SC` is automatically set to the ID of the current dataset, `SPEED_TARGET_ID`.

### *Acceleration limits: skill level*

The acceleration limits used to determine target speed are based on a skill level and aggressiveness limits.

- (14) Drop-down list of three skill levels available for combining lateral and longitudinal acceleration to compare them to specified acceleration limits. These are designated as skill levels 0, 1, and 2:
- 0 Ax and Ay limits are not combined. The target speed is adjusted to allow acceleration either longitudinally or laterally, but never both at once.
  - 1 Ax and Ay are combined using straight lines, allowing some combination of lateral and longitudinal acceleration. However, the combined acceleration does not make use of as much available friction as is used in pure longitudinal or pure lateral acceleration.

- 2 Ax and Ay are combined using a friction ellipse, providing a consistent use of available friction regardless of the direction of the total acceleration vector.

**Note** The methods used to combine lateral and longitudinal acceleration while previewing the path do not consider the detailed dynamics of the vehicle that will occur; they only consider the geometry of the path and the speed to be followed along the path.

Due to dynamic effects, the actual accelerations of the vehicle can exceed the specified limits, especially when aggressive limits are set and wide ranges in speed are simulated (as in a racecourse). This is shown for an example in a following subsection (page 55).

- ⑮ Checkbox to account for road surface 3D geometry. This box is available when lateral and longitudinal acceleration can be combined with skill levels 1 or 2. When checked, the controller considers 3D properties of the road surface along the target path: the banking angle, the grade angle, and the curvature normal to the surface. If not checked, the target speed is calculated by only accounting for the horizontal curvature of the target path.

Some example effects of geometry include:

- The lateral acceleration limit is adjusted based on banking angle. Banking into the turn allows higher speed, while banking out of the turn requires lower speed.
- The longitudinal acceleration limit is adjusted based on vertical road grade. More throttle acceleration is allowed going downhill, while more braking acceleration is allowed going uphill.
- Acceleration limits are adjusted based on curvature normal to the road surface. Higher lateral and longitudinal accelerations are allowed when spring compression increases forces normal to the road surface. Lower lateral and longitudinal acceleration limits must be used when forces normal to the surface are reduced.

### *Acceleration limits: aggressiveness*

Aggressiveness in SC is defined with four limits of acceleration (throttle, braking, left turn, and right turn). Each of the four limits can be specified as either a constant or as a nonlinear Configurable Function. A drop-down list is provided to choose between these options for each component of acceleration (Figure 38).

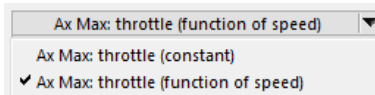


Figure 38. Choose a constant or a link to a Configurable Function screen.

If the constant is chosen, a yellow field appears (e.g., ⑮). If the function is chosen, a data link appears (e.g., ⑮) to the **Generic Table** library. An example function dataset is described later (Figure 40, page 52).

When a Configurable Function is used, it is necessary to specify the keyword associated with the acceleration limit (SPEED\_AX\_THROTTLE, SPEED\_AX\_BRAKE, SPEED\_AY\_LEFT, or SPEED\_AY\_RIGHT). Right-click on the drop-down control to see the keyword associated with that control. For example, right-click on the throttle pull-down control to see that the keyword is SPEED\_AX\_THROTTLE (Figure 39).

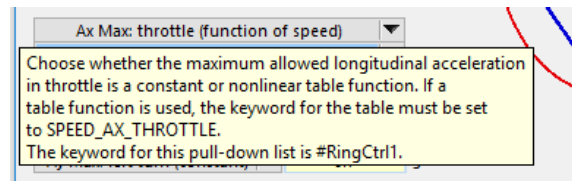


Figure 39. Right-click to see the root keyword for a configurable function.

Each of the four Configurable Functions has six associated parameters available to transform the three variables involved in the function: there are gains and offsets for station, speed, and acceleration limit. These are described in the subsection **Transforming an Acceleration Limit Shaping Function** (page 53).

- ①⑥ Limit longitudinal acceleration allowed during application of throttle. The limit can be specified as either a constant (keyword = SPEED\_AX\_THROTTLE\_CONSTANT) or as a nonlinear Configurable Function of speed and station (root keyword = SPEED\_AX\_THROTTLE).
- ①⑦ Limit longitudinal acceleration allowed during braking. The limit can be specified as either a constant (keyword = SPEED\_AX\_BRAKE\_CONSTANT) or as a nonlinear Configurable Function of speed and station (root keyword = SPEED\_AX\_BRAKE).
- ①⑧ Limit lateral acceleration allowed during a left-hand turn. The limit can be specified as either a constant (keyword = SPEED\_AY\_LEFT\_CONSTANT) or as a nonlinear Configurable Function of speed and station (root keyword = SPEED\_AY\_LEFT).
- ①⑨ Limit lateral acceleration allowed during a right-hand turn. The limit can be specified as either a constant (keyword = SPEED\_AY\_RIGHT\_CONSTANT) or as a nonlinear Configurable Function of speed and station (root keyword = SPEED\_AY\_RIGHT).

### Path preview lengths

Previewing of the target path by SC is configured with four length parameters.

- ②⑩ Length of path segment used to calculate curvature at the mid-point of the segment (keyword = SPEED\_CURV\_LENGTH). The controller calculates curvature considering the geometry of the reference path plus a lateral offset target described previously (see Figure 19 on page 26). If a short length is used, the controller will sense short sections of high curvature and reduce the target speed accordingly. If a long length is used, the controller will only reduce speed for sustained turns. A typical value for a road would be 50 m.

The specified segment length is rounded off internally to an integer multiple of the preview interval ②③.

- ②① Start of path preview (`SPEED_PREVIEW_START`). The portion of the reference path that is previewed starts this distance in front of the origin of the vehicle sprung mass coordinate system (typically the origin is at the center of the front axle). Setting the starting point in front of the vehicle compensates somewhat for the dynamic lag in the vehicle response.

The specified distance is rounded off internally to an integer multiple of the preview interval ②③.

- ②② Total length of path preview (`SPEED_PREVIEW`). This defines the portion of the reference path that is previewed. It should be at least the distance needed to slow from the fastest allowed speed to the slowest speed at the specified limit acceleration. Longer distances sometimes give better results for complicated paths combined with aggressive acceleration settings.

The specified distance is rounded off internally to an integer multiple of the preview interval ②③.

- ②③ Interval for calculating path curvature and target speed over the preview path (`SPEED_PREVIEW_STEP`). The other three lengths are rounded off internally to integer multiples of this value. Hence, it defines the resolution of the other lengths. A typical value is 1 or 2 m.

<b>Note</b>	SC maintains a few internal arrays for speeds and accelerations for every preview point. The number of preview points is $\text{SPEED\_PREVIEW} / \text{SPEED\_PREVIEW\_STEP} + 1$ . If this number is very large (more than a few hundred, as might occur if the interval is small), the simulation will run noticeably slower.
-------------	--

### *Additional controls*

- ②④ Checkbox to show two optional controls: Misc. yellow field and Misc. link.
- ②⑤ Miscellaneous field. Use this to add VS Commands, including additional information related to SC.
- ②⑥ Miscellaneous link. Use this to add information for use by SC.

### *Using the Generic Table screen*

If one or more of the acceleration limits (①⑥ - ①⑨) are specified as functions of speed, then links are made to the **Generic Table** library. This screen (Figure 40) can be used to represent almost any Configurable Function used within a VS Math Model. In the figure, it is used to represent the throttle acceleration limit in SC.

Keywords are used to identify the function.

- ① This comment field identifies the variable whose value is calculated by the Configurable Function. This text is not sent to the VS Math Model, but is helpful for documenting the contents on the screen.

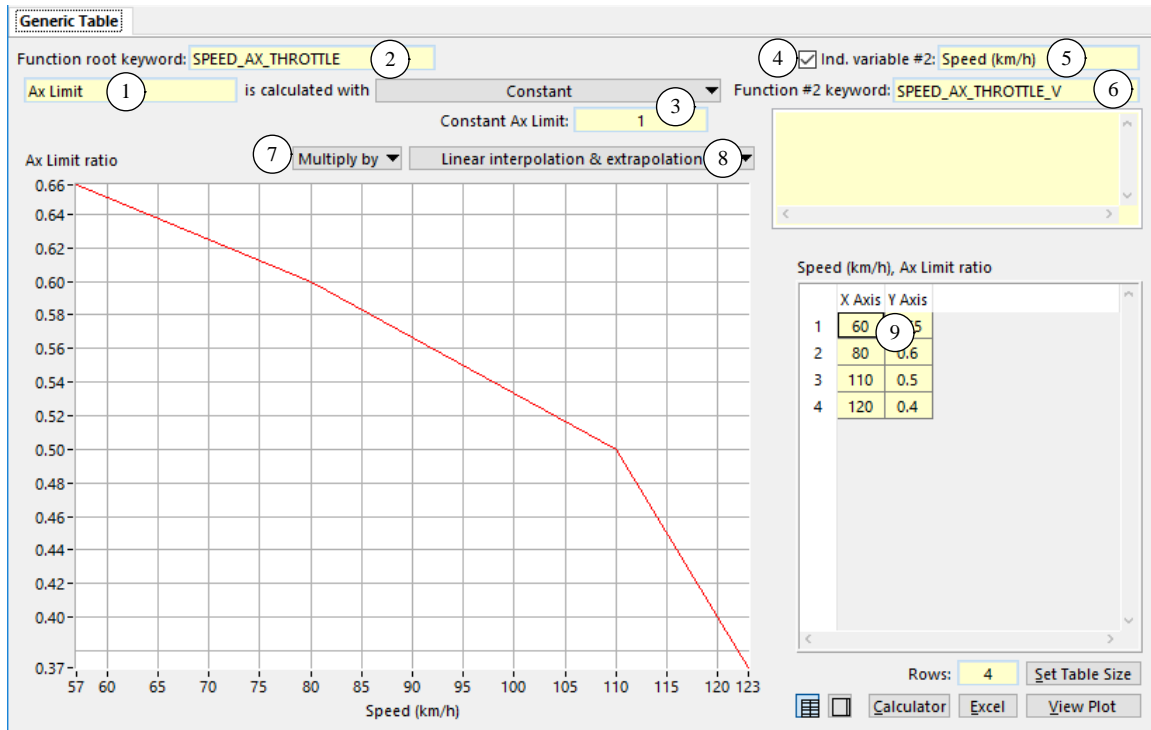


Figure 40. Defining an acceleration limit as a function of speed and station.

- ② The root keyword for the function must be set for the math model to assign the data properly to the Configurable Function. In general, Echo files are a good reference for keywords because the root keywords are written in the Echo files created for each run. In the case of the Speed Controller from Path Preview, the keyword can also be found by right-clicking on the pull-down control for the limit acceleration (Figure 39 on page 50).

The keyword written in the parsfile that is read by the VS Math Model is automatically extended, based on the type of calculation that is selected ③, as described below.

- ③ Pull-down control to specify type of calculation (constant, linear interpolation, 2D table, etc.) associated with the primary variable. In this example, there is no sensitivity to station; instead, the limit is a function of vehicle speed. Therefore, the constant option is selected and a value of 1 is specified such that it can be multiplied as a function of speed.

With these selections, the keyword written into the parsfile that is sent to the VS Math Model is `SPEED_AX_THROTTLE_CONSTANT` (Figure 41).

- ④ Checkbox to indicate that there is a secondary independent variable (speed).
- ⑤ Comment field that identifies the secondary variable for the Configurable Function. This text is not sent to the VS Math Model but instead documents the contents on the screen.
- ⑥ Root keyword for the second function used to calculate the effect of the second independent variable.
- ⑦ Drop-down control that specifies whether two functions are added or multiplied.

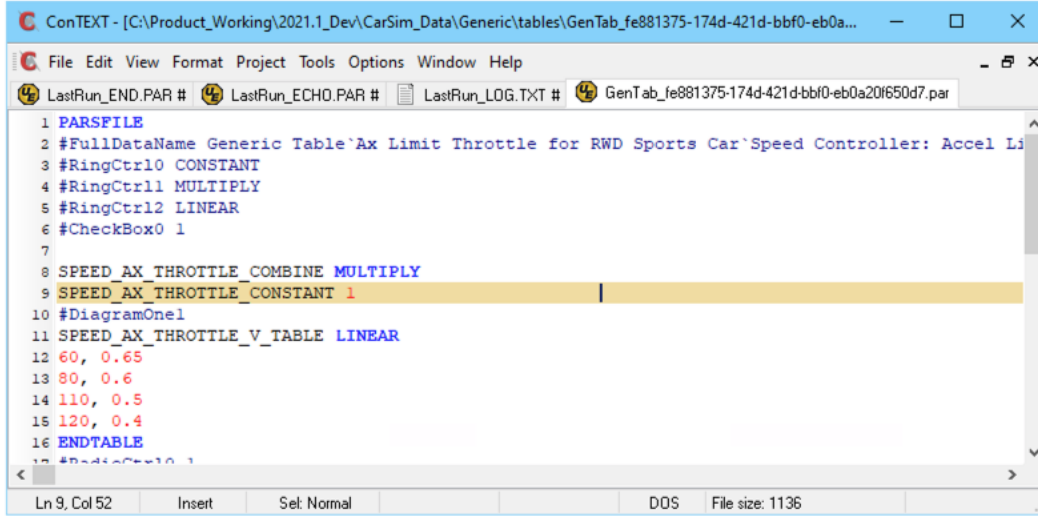


Figure 41. Parsfile for the Configurable Function specified in Figure 40.

- ⑧ Drop-down control to specify a type of calculation (constant, linear interpolation, etc.) associated with the second variable; in this case, vehicle speed.

With the selections shown in the figure, the line of text written into the parsfile that is sent to the VS Math Model is `SPEED_AX_THROTTLE_V_TABLE LINEAR`.

- ⑨ Tabular data used to calculate the acceleration limit.

### Transforming a Configurable Function

Each of the four Configurable Functions used to define acceleration limits includes six parameters that can be used to transform the three variables that are related by the function. This is similar to the scaling available for the open-loop controls described earlier (page 4), but in this case, the transforms include another independent variable.

$$acceleration\_limit = f(V, S) \cdot gain + offset$$

$$acceleration\_limit = f([v_X - vstart]/vscale, [station - sstart]/sscale) \cdot gain + offset \quad (18)$$

where

1.  $f$  is a function that uses a method you define on the screen (e.g., 2D linear interpolation is specified in Figure 40).
2.  $S$  and  $V$  are the independent variables (speed and station) shown on the screen and used to calculate  $f$ ;  $S$  is in turn defined as a function of station and two parameters  $sstart$  and  $sscale$ , and  $V$  is defined as a function of vehicle speed and two parameters  $vstart$  and  $vscale$ ;
3.  $gain$  is a dimensionless multiplier applied to the function; and
4.  $offset$  is an offset applied to the function.

Table 8 shows the naming conventions for the parameters of the four configurable functions that define acceleration limits for SC, corresponding to the six parameters referenced in equation 18. In the table, *type* can be `AX_THROTTLE`, `AX_BRAKE`, `AY_LEFT`, or `AY_RIGHT`.

Table 8. Naming convention for keywords for acceleration limit configurable functions.

Component	Keyword	Example for <i>type</i> = <b>AX_THROTTLE</b>
<i>function</i>	SPEED_ <i>type</i>	SPEED_AX_THROTTLE
<i>gain</i>	SPEED_ <i>type</i> _GAIN	SPEED_AX_THROTTLE_GAIN
<i>offset</i>	SPEED_ <i>type</i> _OFFSET	SPEED_AX_THROTTLE_OFFSET
<i>sstart</i>	SSTART_ <i>type</i>	SSTART_SPEED_AX_THROTTLE
<i>sscale</i>	SSCALE_ <i>type</i>	SSCALE_SPEED_AX_THROTTLE
<i>vstart</i>	VX_START_ <i>type</i>	VX_START_SPEED_AX_THROTTLE
<i>vscale</i>	VX_SCALE_ <i>type</i>	VX_SCALE_SPEED_AX_THROTTLE

Data from the Generic Table screen (Figure 40) can be treated as a shaping function and transformed, typically by setting the scaling parameters in the Misc. yellow field on the **Control: Speed (Closed Loop) Using Path Preview** screen ((25), Figure 37, page 47). For example, set  $VX\_SCALE\_SPEED\_AX\_THROTTLE = 1.2$  to multiply the range of speeds covered in the table by 1.2, to apply an existing acceleration limit for a vehicle with a higher performance engine.

The same transforms are available for the SPEED\_TARGET Configurable Function, whose independent variables are time and station. Table 9 lists the six keywords used to specify offsets and gain (scale) for the calculated speed and the two independent variables.

Table 9. Keywords for the SPEED\_TARGET configurable function.

Component	Keyword
<i>function</i>	SPEED_TARGET
<i>gain</i>	SPEED_TARGET_GAIN
<i>offset</i>	SPEED_TARGET_OFFSET
<i>sstart</i>	SSTART_SPEED_TARGET
<i>sscale</i>	SSCALE_SPEED_TARGET
<i>tstart</i>	TSTART_SPEED_TARGET
<i>tscale</i>	TSCALE_SPEED_TARGET

### *Example: target speed on a racetrack*

The original goal of the path preview mode (OPT\_SC = 4) was to look ahead on a racecourse and determine a target speed that would be mainly limited by tire friction. Figure 42 shows some simulation results on a racetrack in CarSim, running with path preview. The SC dataset for this example is the one used to document the controls on the screen (Figure 37, page 47).

In this example, the maximum speed is set to 150 km/h, and aggressive acceleration limits were set: 0.7g lateral, 1.0g braking, and speed dependent for accelerating (based on engine capability, using the example shown in Figure 40, page 52). Road geometry is also considered.



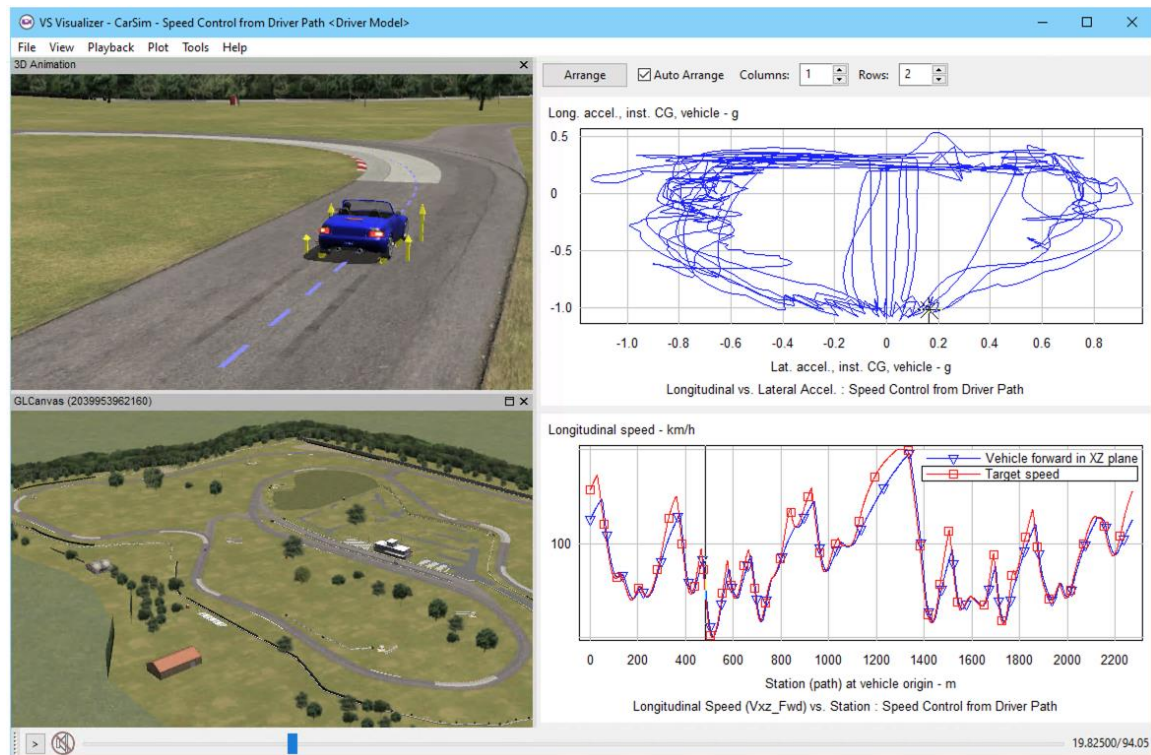


Figure 42. Speed target set for racetrack performance.

The speed plot shows that upper speed limit of 150 km/h was used as a target speed for a only a very small interval, at a station of about  $S=1300\text{m}$ . The actual vehicle speed never quite reached this limit.

Notice in the acceleration cross-plot that the lateral limits are sometimes exceeded, as is the braking acceleration limit. As noted earlier, this is expected. The calculations of target acceleration use a simplified internal model without all the dynamic detail of the full vehicle model.

### Example: target speed for a $90^\circ$ turn

Figure 43 shows how SC handles a  $90^\circ$  turn when a traffic light is green during the approach and start of the turn. SC creates tables of variables, including target speed, as functions of station ahead of the current vehicle position. The plot on top shows the target speed and vehicle speed plotted against station. In this example, the acceleration limits were modest, and the brakes and powertrain were sufficient to have the vehicle speed follow the target speed closely.

Although SC provides target speed as a function of station, it uses time-based calculations to determine accelerations and speeds. The bottom set of plots in the figure show a straight line for deceleration from 1s to 8s (constant deceleration), and another straight line for accelerating (constant acceleration) after the turn, from 13s to 25s.

During the first part of this simulation (up to about 12s into the run), the upper speed limit was a constant of 45 mi/h (the units for speed were changed from km/h to mi/h to match speed limit signs in the USA). At about 12s, the target speed was changed to 55 mi/h due to the detection of a speed limit sign. All the changes in speed shown in the plots were solely to the presence of the  $90^\circ$  turn.



Figure 43. Plots showing speed control for a 90° turn at an intersection.

### Changing Path\_ID\_DM or LTARG\_ID\_DM with Events

When SC operates in path preview model ( $OPT\_SC = 4$ ), the path preview is based on the path identified with  $PATH\_ID\_DM$  and  $LTARG\_ID\_DM$ . These are used even if DM is not used ( $OPT\_DM = 0$ ), e.g., when steer control is from external software.

If either of the parameters  $PATH\_ID\_DM$  or  $LTARG\_ID\_DM$  are changed via an Event, SC will reset to use the new path for the preview.

### Vehicle Tests with Many Changes in Speed Target

Everything about speed control can be changed at any time during a simulation by triggering a VS Event that specifies new settings. The only requirement is that if SC will be used, it must be installed before the simulation starts, as described earlier (page 32).

Simulations involving complicated vehicle tests might require multiple  $SPEED\_TARGET$  datasets.

If there are no moving objects, then only one  $SPEED\_TARGET$  dataset is in use at a time and is identified with the SC parameter  $SPEED\_ID\_SC$ . Changes in the target speed can be made from an Events dataset by using the built-in controls shown earlier (Figure 35) or by entering text information in a miscellaneous field.

If an **Events** dataset is used only once, then the built-in controls are probably the easiest to use. It is not necessary to use a custom ID, although there is no harm in doing so.

However, if an **Events** dataset might be used multiple times, then there is an advantage to using a custom ID. Doing so will mean a single  $SPEED\_TARGET$  dataset will be used every time the **Events** dataset Parsfile is read by the VS Math Model. The first time, it will create a new  $SPEED\_TARGET$  dataset and set the SC parameter  $SPEED\_ID\_SC$  to match it. Every subsequent reference will simply set the SC parameter to the ID of the existing dataset.

For example, the **Events** dataset shown in Figure 35 (page 43) is used multiple times in a test procedure called “Sine with Dwell.” The first part of the repeated test specifies that the vehicle speed is to be brought up to 82 km/h and then allowed to coast down to 80 km/h. The example **Events** screen sets the target speed to 83 km/h, causing acceleration. Because this part of the test is repeated many times, a custom ID is set such that the same SPEED\_TARGET dataset, with ID 1083, will be reused after the first time.

Another option is to not use the built-in controls. Instead, choose the option on the speed control drop-down **Do not specify speed control here**, and set the SPEED\_TARGET properties in a miscellaneous field. For example, Figure 44 shows part of the **Events** screen for a dataset used in a vehicle test that involves driving in a constant-radius turn at a constant speed until quasi-static equilibrium as obtained, and then repeating at a higher speed. (Several examples are provided in the CarSim and TruckSim databases, in the category **Handling and Stability Tests**, involving the **ISO Steady-State Circle** test.) In this example, a new target speed is calculated from a target lateral acceleration and assigned to the Configurable Function using the SPEED\_TARGET\_CONSTANT keyword.

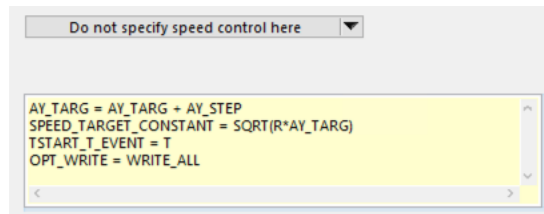


Figure 44. Setting speed target using a miscellaneous yellow field.

If the speed target is a constant, whose value should be changed using an **Events** dataset, it is often easiest to set a new value in a miscellaneous field using the SPEED\_TARGET\_CONSTANT keyword, as shown in the figure.

## Speed Control for Starting, Stopping, and Reversing

SC has existed almost as long as CarSim and TruckSim. It was originally intended to support standard vehicle dynamics testing that involves running at a constant speed, or basic highway driving conditions. Options were added to use Configurable Functions for target speed based on time or station. The path preview option was added to support limit-based control on racetracks, and normal driving on public roads. More recently, pure acceleration control was added as a target for the controller.

Many of the ADAS examples that have been provided with CarSim and TruckSim have involved transient scenarios, such as stopping for a stop sign or traffic signal, restarting when the traffic and crosswalks are clear, or changing “plans” based on traffic signs and other situations.

As shown from these examples, the existing capabilities of SC can be used to support many scenarios beyond the simple conditions of steady speeds or simple vehicle testing.

### Looking for signs and traffic lights

There are several ADAS examples in CarSim and TruckSim that have the vehicle brake to a stop at a traffic light or stop sign. They usually have a simulation of the driver monitoring the environment with pending Events for signs (Figure 45).

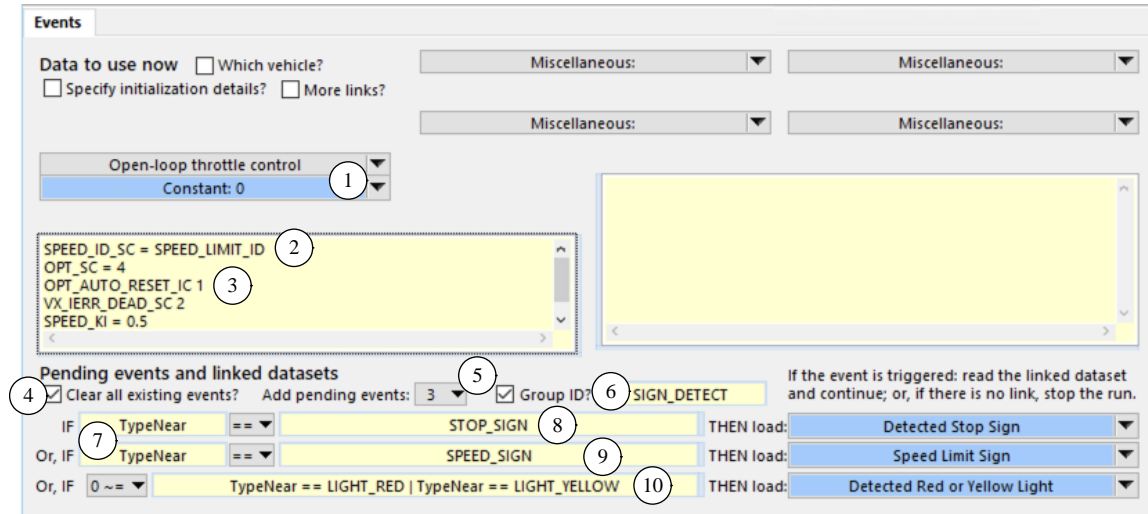


Figure 45. Setting up Events to handle signs and traffic lights.

The dataset sets up several things:

1. The open-loop throttle is set to zero (1). This is done in case open-loop throttle was in use when this Event was triggered.
2. SPEED\_ID\_SC (the SPEED\_TARGET ID used by SC) is set to SPEED\_LIMIT\_ID, a parameter defined by a VS Command elsewhere (2).
3. Several SC parameters are set (3). These include OPT\_SC, OPT\_AUTO\_RESET\_ID, VS\_IERR\_DEAD\_SC, and SPEED\_KI.
4. Three new Events are added (5), in a group identified with a parameter SIGN\_DETECT that was defined by a VS Command elsewhere (6). Any existing Events in this group are cleared when this dataset is loaded (4).
5. All three new Events are based on a variable TypeNear, defined by a VS Command elsewhere. This is a property of a Target Object detected by an ADAS sensor looking for traffic signs.
6. One of the Events is triggered if TypeNear == STOP\_SIGN (8), where STOP\_SIGN is a parameter defined by a VS Command elsewhere.
7. A different Event is triggered if TypeNear == SPEED\_SIGN (9), another parameter defined by a VS Command elsewhere.
8. A different Event is triggered if TypeNear is either LIGHT\_RED or LIGHT\_YELLOW (10), two more parameters defined by VS Commands elsewhere.

### Stopping at a sign or traffic light

If an Event is triggered to cause the vehicle to stop for a stop sign or red light, the speed target dataset might be switched to a normalized speed dataset (Figure 46).

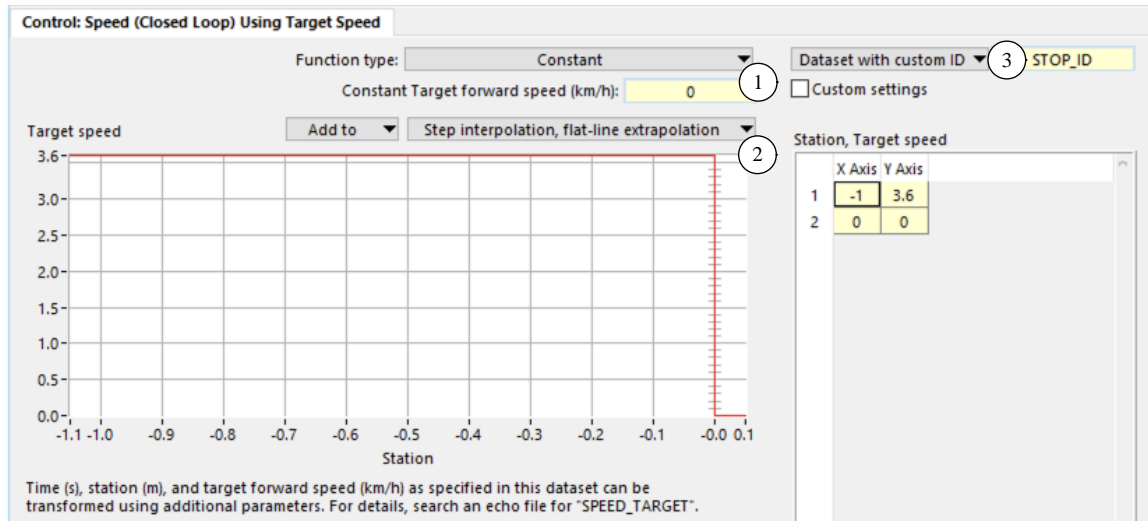


Figure 46. Normalized target speed dataset with 1 m/s (3.6 km/h) constant for  $S < 0$ , then 0.

This dataset shows no influence of time on the `SPEED_TARGET` output (1), but does define a step function related to station (2):

1. For station  $S < 0$ , the target is a constant 3.6 km/h, which is 1 m/s.
2. For station  $S \geq 0$ , the target is zero speed.

If this dataset were used “as is,” with no scaling or offsets, SC would maintain a constant speed of 1.0 m/s until the preview detects a need to reach a target speed of 0, at  $S = 0$ . Deceleration would be targeted to match the limit specified on the Path Preview screen.

This dataset includes a user ID, assigned to parameter `STOP_ID` (3) that was defined by a VS Command elsewhere.

The dataset is not used “as is.” It is intended to be activated by an Event, as shown earlier in Figure 45. For example, Figure 47 shows the Event dataset that was specified in Figure 45.

This dataset does nothing to change SC with the built-in controls (1), but it does cause the VS Math Model to make some adjustments:

1. The current `SPEED_TARGET` index `ISPEED` is set, based on the parameter `STOP_ID` via the VS Command `SET_ISPEED_FOR_ID` (2). This `SPEED_TARGET` dataset is the one shown in Figure 46.
2. The vertical (speed) scale `SPEED_TARGET_GAIN` is set to the current vehicle speed, `Vxz_Fwd` (3). Multiplying this scale with the original value of 1 m/s means the `SPEED_TARGET` dataset now has an initial value equal numerically to the current value of `Vxz_Fwd`.



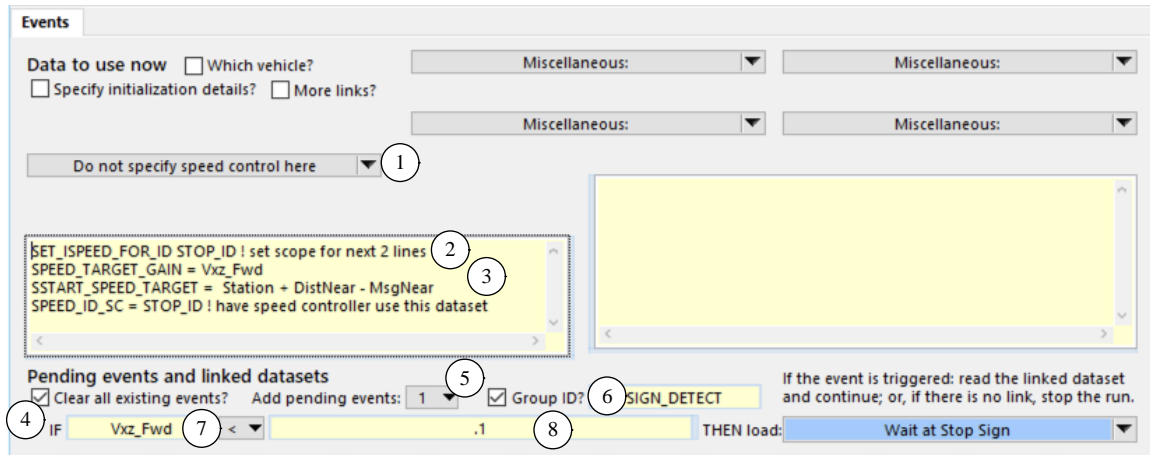


Figure 47. Example Event that is triggered when a stop sign is detected.

3. The independent variable  $S$  (station) is offset such that the target speed drops to zero at a value of  $S$  related to a stop sign or traffic signal. This is done by setting the value of  $SSTART\_SPEED\_TARGET$  to the current vehicle station ( $Station$ ) plus the distance to the detected stop sign ( $DistNear$ ), minus the distance from the sign to the stopping point ( $MsgNear$ ). In this example, the output variables  $DistNear$  and  $MsgNear$  were defined elsewhere.
4.  $SPEED\_ID\_SC$  is also set to  $STOP\_ID$ .

Any pending Events with Group ID  $SIGN\_DETECT$  are cleared, and another pending Event is defined. The new pending Event will be triggered when the forward speed  $Vxz\_Fwd$  drops below 0.1 m/s. When that occurs, another **Events** dataset is read with conditions for watching the intersection until it clears (Figure 48).

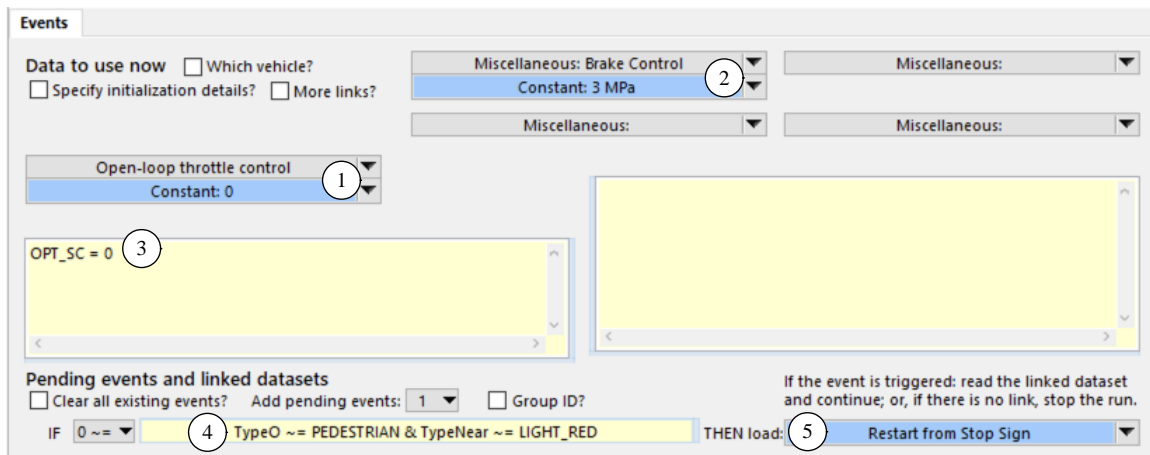


Figure 48. Waiting at an intersection until it is clear to go.

In this state of waiting,  $SC$  is off, the open-loop throttle is 0, and the brakes are applied. A pending Event is defined to restart when the intersection is clear and there is not a red light. (This Event is used for both stop signs and traffic lights.)

Figure 49 shows a camera view and speed plots for a simulation that uses the above examples for TARGET\_SPEED and the stopping Event. The conditions defined by the **Events** dataset in Figure 47 apply from  $T = 0.5s$  to  $T = 10.3s$  (Station = -110m to Station = -12m).

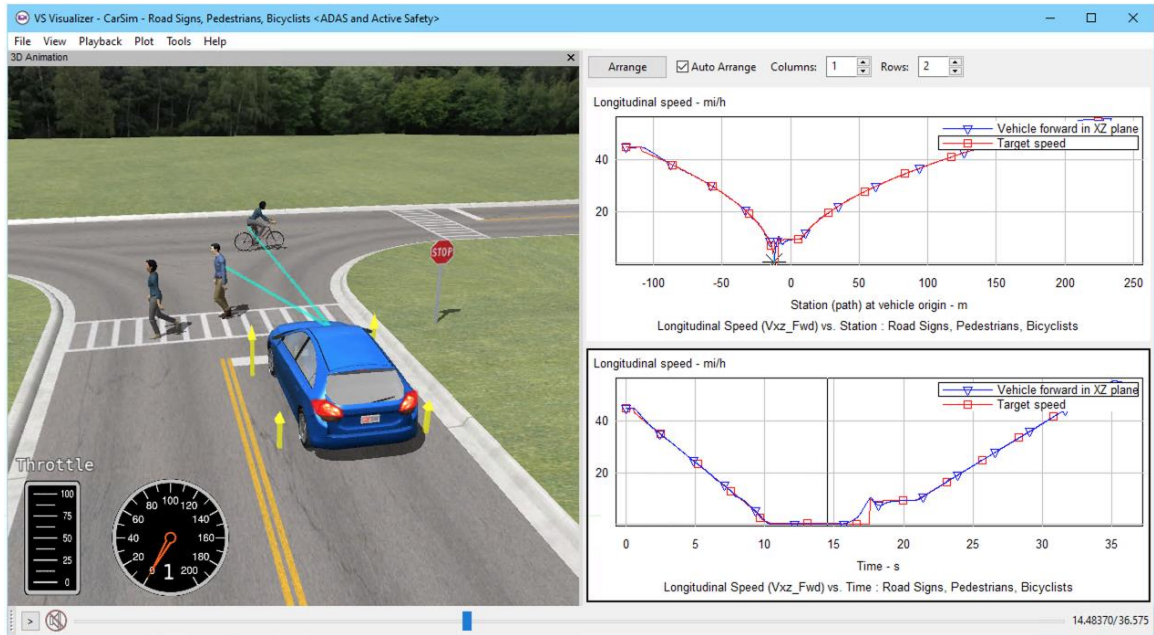


Figure 49. Plots showing SC stopping at a stop sign, and then continuing when clear.

### Starting from a stop sign or traffic light

The plots shown in Figure 49 indicate that the vehicle starts from stop at about 16s into the run. From Figure 48, we see that this occurred when an Event was triggered in the case where there were no target objects of type PEDESTRIAN in view. (In this example, the bicyclist moving object also has type PEDESTRIAN.)

We have found that when starting from a stopped condition, the speed controller in a path preview mode will start too abruptly, trying to achieve the limit acceleration. One way to improve this is to have a speed-dependent acceleration limit. Another way, used in this example, is to start with open-loop throttle using a smooth transition (Figure 50). This dataset for the throttle control is defined with just three data points (2) going from 0 to 0.2 throttle over an interval of 2s, using the Configurable Function calculation method **Spline interpolation, flat-line-extrapolation** (1).

Figure 51 shows the **Events** dataset that is loaded when the intersection is clear (as indicated in Figure 48).

When loaded, the brakes are released (2), and the open-loop throttle control from Figure 50 is activated. Notice that the throttle time history starts at  $T = 0$  in Figure 50. We want Time = 0 in Figure 50 to match the current simulation time  $T$ . This is done by setting a parameter in the throttle table TSTART\_THROTTLE to  $T$  (3). This **Events** dataset also adds a pending event that will be triggered when the vehicle speed reaches 10 km/h (4). When this occurs, the VS Math Model will read the Parsfile for the **Events** dataset **Look for Traffic Signals** (5), which was already shown in Figure 45 (page 58). That activates SC again, with the settings described earlier.



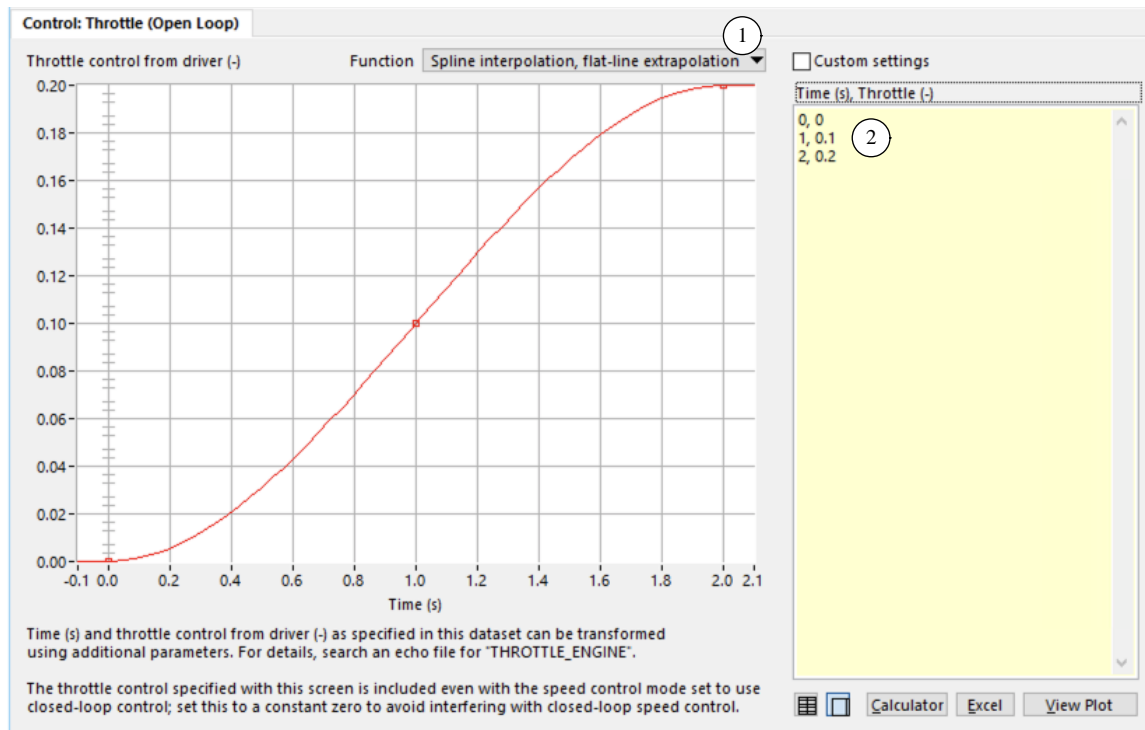


Figure 50. Open-loop throttle to start smoothly from a stopped condition.

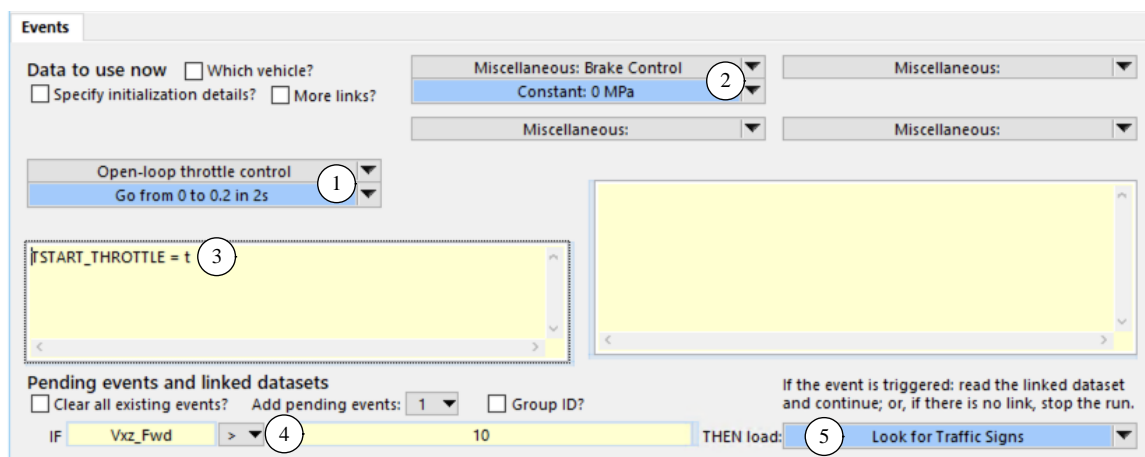


Figure 51. Event to start from a stopped condition.

## Shift Control

If the vehicle powertrain includes a transmission with discrete gears (i.e., not CVT), the transmission gear can be specified directly as a driver control or determined with a built-in closed-loop controller that can mimic a driver or represent shifting in an automatic transmission. Settings for control of the transmission gear are made with datasets from one of two libraries:

1. **Control: Shifting (Closed Loop).** Datasets in this library set up a mode for shifting automatically using a built-in closed-loop controller.

2. **Control Shifting (Open Loop).** Datasets in this library set the mode to open-loop and set up a Configurable Function that sets the gear to an open-loop function of time.

The following values are allowed for gear numbers:

- 1 Reverse.
- 0 Neutral.
- 1-18 Forward gear. The upper limit is the number of gears NGEARS, which in turn has a built-in limit of 18.

## Specifying Closed-Loop Control of Transmission Gear

The closed-loop controller option is specified by a Configurable Function `MODE_TRANS` that can be set to a constant, equation, or a step table using the **Control: Shifting (Closed Loop)**, shown in Figure 52. The valid values of `MODE_TRANS` are:

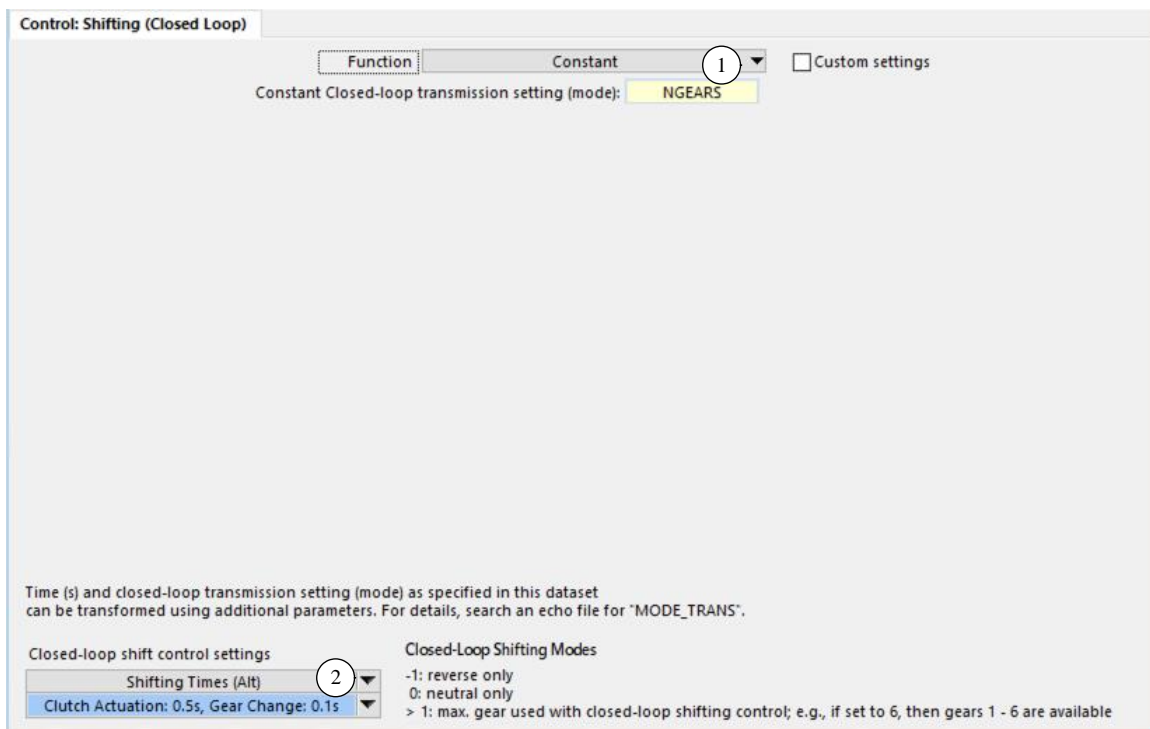


Figure 52. The Control: Shifting (Closed Loop) screen.

- 1 The vehicle is in reverse.
- 0 The vehicle is in neutral.
- 1 The transmission gear is specified with the open-loop function `GEAR_TRANS`, which is described in the next subsection.
- 2-18 Shifting is handled by the closed-loop controller using shift tables. Although the VS Math Models support a maximum value of 18, it is limited by the number of gears in the transmission `NGEARS`, which has a value between 1 and 18.

If the transmission has a clutch, then the blue link ② supports connection to a closed-loop clutch controller with settings for synchronizing gear shifts with clutch disengagement and throttle modulation. This option is described in a later subsection (page 65).

As noted above, `MODE_TRANS` is a Configurable Function. It is often set to a constant (with keyword `MODE_TRANS_CONSTANT`) if the same shifting mode is used throughout a simulation run.

Figure 53 shows the parameters in the VS Math Model that also affect the closed-loop shifting control. Two of these parameters are not set automatically by any of the VS Browser screens: `LIMIT_DOWNSHIFT` and `LIMIT_UPSHIFT` (lines 783 and 784). When left at the default values of 1, the internal shifting uses a single set of upshift and downshift Configurable Function datasets to consider shifting one gear up or down. When set to a higher number (e.g., `LIMIT_UPSHIFT` = 3), the controller will check the appropriate shift Configurable Function dataset for the largest shift possible, subject to the limit that the highest gear allowed is `NGEARS` (line 785) and the lowest gear is 1. If the combination of throttle and transmission speed would indicate a shift (e.g., upshift from gear 3 to 7 if `LIMIT_UPSHIFT` = 3), then a shift is made. If not, the controller attempts the next closer gear (e.g., upshift from 3 to 6). If not successful, it goes to the next closer gear (e.g., from 3 to 5) and so on until all shifts from the limits down to 1 have been tested.

```

767 !-----
768 ! TRANSMISSION
769 !-----
770 ! The transmission is specified with the following parameters and Configurable
771 ! Functions noted below. Transmission controller mode can be specified with the
772 ! open-loop function MODE_TRANS and transmission gear can be specified with the
773 ! open-loop function GEAR_TRANS.
774
775 INSTALL_TRANSMISSION ! VS Command to install a transmission
776
777 OPT_TRANS_INTERNAL 1 ! Transmission model: 1 -> internal, 0 -> external [I]
778 OPT_TR_GEAR_INTERNAL 1 ! Transmission gear ratio and inertia: 1 -> up to 18 gears, 2
779 ! -> continuously variable (CVT), 0 -> external model.
780 ! Option 1 uses functions DOWNSHIFT_TRANS and UPSHIFT_TRANS.
781 ! Option 2 uses functions R_GEAR_CVT, R_EFF_CVT_F, and
782 ! R_EFF_CVT_R. [I]
783 LIMIT_DOWNSHIFT 1 ! [D] Limit to number of gears covered in a downshift
784 LIMIT_UPSHIFT 1 ! [D] Limit to number of gears covered in an upshift
785 NGEARS 7 ! Number of forward gears in transmission [I]
786 OPT_SHIFT_INTERNAL 1 ! Gear shift command model: 1 -> internal, 0 -> external [I]
787

```

Figure 53. Settings for shift control in a transmission.

All the parameters shown in Figure 53 are set by connecting with transmission screens, with the exceptions of `LIMIT_DOWNSHIFT` and `LIMIT_UPSHIFT`. These are not built into the transmission screens and must be set using miscellaneous yellow fields.

## Specifying Transmission Gear with an Open-Loop Function

When the open-loop mode is in effect (`MODE_TRANS` = 1), the gear shift position (keyword = `GEAR_TRANS`) is specified as an open-loop function of time using datasets from the library **Control: Shifting (Open Loop)**, shown in Figure 54.

Every dataset from this library sets `MODE_TRANS` = 1 and provides a dataset for the `GEAR_TRANS` Configurable Function. Given that gears are integer numbers, the only options for setting the function type are Constant, Step, and Equation ①.

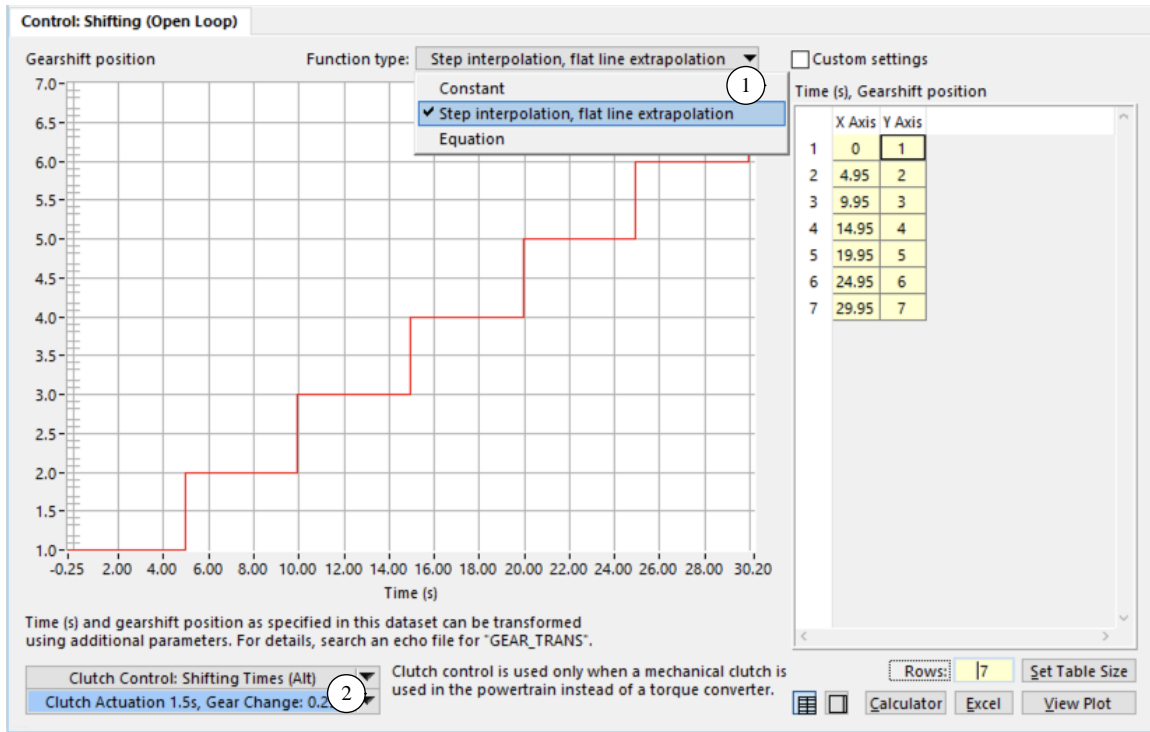


Figure 54. The Control: Shifting (Open Loop) screen.

**Note** The Reverse and Neutral settings for the transmission may be set by two methods. If `MODE_TRANS` is set to -1 or 0, the transmission is in reverse or neutral, respectively, regardless of the value of `GEAR_TRANS`. Alternatively, if `MODE_TRANS` is set to 1, then `GEAR_TRANS` may be set to -1 for reverse or 0 for neutral.

If the transmission has a clutch, then the blue link (2) supports connection to an open-loop clutch vs. time, or a closed-loop controller with settings for synchronizing gear shifts with clutch disengagement and throttle modulation.

## Closed-Loop Shifting by Clutch

The powertrain may be set up to use a representation of a hydraulic torque converter or a mechanical clutch. If a clutch is used, a control signal is used to release and engage the clutch.

There are two modes for specifying the clutch control signal.

1. Open-loop clutch control, as specified with the **Control: Clutch (Open Loop)** screen and/or imported from Simulink or other external code.
2. A standard sequence can be applied whenever a shifting of the transmission gear occurs. The sequence is defined with parameters that are set from the **Clutch: Clutch Shifting Timelines (Closed Loop)** screen (Figure 55).

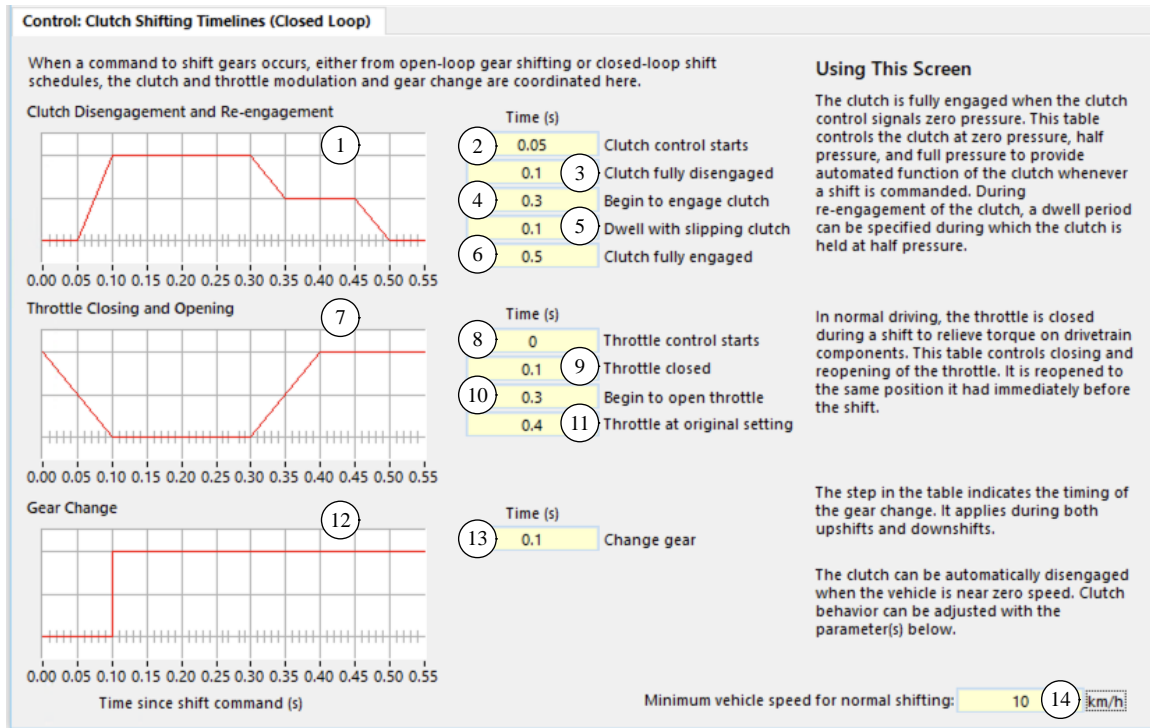


Figure 55. Screen with closed-loop clutch shifting timelines.

The clutch control mode is selected by linking to a control screen, and the information is passed to the math models with the keyword `OPT_CLUTCH_MODE` with a value of 0 for open loop or a value of 1 for the standard built-in sequence. (The closed-loop screen shown in Figure 55 always sets `OPT_CLUTCH_MODE = 1`.)

Because the clutch control is tightly associated with gear shifting, it is normally linked to the shift control screen, which is either closed loop (**Control: Shifting Timelines (Closed Loop)**) or open loop (**Control: Shifting (Open Loop)**).

The **Control: Clutch Shifting Timelines (Closed Loop)** screen uses a timeline concept to define the events involved in a gear shift, where the timeline starts when a command to shift is received by the controller.

When a command is triggered to shift, the controller closes the throttle and applies pressure to the clutch, raising the clutch control to unity (fully disengaged). While the clutch is disengaged, the gear ratio is changed. Then, the clutch pressure is released halfway (the control is dropped to 0.5, or half engaged) and held for a specified time, and then released the rest of the way to zero (fully engaged).

The plots on this screen are updated “live” to show how all the specified control changes are related in time using the current values of the parameters in the adjacent yellow fields. Each plot begins at the moment a shift is commanded, either by open loop shifting control or by shifting based on closed-loop upshift and downshift schedules.

The same sequence applies to both upshifts and downshifts.

- ① This section controls engagement and disengagement of the clutch. The plot shows how the control signal is modulated between zero (fully engaged) and one (fully disengaged).
- ② Time from the moment a gear shift command occurs until the clutch starts to disengage (keyword = T\_CL\_START).
- ③ Time from the moment a gear shift command occurs until the clutch is fully disengaged (keyword = T\_CL\_DISENGAGE). Disengagement of the clutch begins at the instant the shift command occurs. Normally the clutch is fully disengaged before the gear change occurs.
- ④ Time after the gear shift command until the clutch begins to re-engage (keyword = T\_CL\_PRESS\_TOTAL). Normally this is after the gear change.
- ⑤ The clutch control can be made to dwell at half-engagement (keyword = T\_CL\_HALF\_HOLD). Depending on the load on the drivetrain, the throttle setting, and the clutch torque as a function of the control signal, the clutch may slip at this setting, producing a smoother engagement. If you prefer that the clutch does not dwell at half-engagement, enter zero in this field. Unlike all the other times specified on this screen, this value defines the duration of the dwell time, not the time of its occurrence. It is always placed in the middle of the clutch re-engagement.
- ⑥ Time after the shift command occurs until the clutch is once again fully engaged (keyword = T\_CL\_RE\_ENGAGE).
- ⑦ This section controls closing and opening of the throttle. The plot shows how the throttle is modulated between the setting immediately prior to the shift and zero throttle. After the shift, the throttle is returned to the setting prior to the shift, and throttle control is returned to either control via open-loop or closed-loop settings.
- ⑧ Time from the moment a gear shift command occurs until the throttle request starts to close (keyword = T\_TH\_START).
- ⑨ Time after the shift command until the throttle is completely closed (keyword = T\_TH\_ZERO). The throttle begins to close at the instant the shift command occurs. Normally this is before the gear change.
- ⑩ Time after the shift command until the throttle begins to open again (keyword = T\_TH\_ZERO\_TOTAL). Normally this is after the gear change.
- ⑪ Time after the shift command until the throttle is returned to the setting it had prior to the shift command (keyword = T\_TH\_RETURNED). The After this point, the throttle control is returned to whatever control (e.g., open-loop tables or closed-loop speed control) that was in use prior to the shift.
- ⑫ This section controls the timing of the actual ratio change. When the shift command occurs, the ratio change is made a time that allows the clutch to be disengaged and the throttle to be closed. The step in the table only indicates the timing of the ratio change. It applies during both upshifts and downshifts.
- ⑬ Time after the shift command when the ratio change occurs (keyword = T\_GEAR\_LAG).



- ⑭ Low speed for automatically disengaging the clutch (keyword = VLOW\_CLUTCH). When braking, the clutch might be disengaged when the average wheel speed drops below this speed. When accelerating from a low starting speed, the clutch will initially be disengaged if the starting speed is below this value and will be engaged according to the other settings on this screen. If the starting speed is above this speed, the clutch is engaged as the run starts.

Unlike most of the SC parameters mentioned in this document, the closed-loop clutch parameters may be used to represent built-in behavior of powertrains with mechanical clutches even if the SC is not installed.

They are listed with other powertrain settings in the Torque Transfer Device section of the Echo file (lines 743 – 765, Figure 56).

```

722 ! -----
723 ! TORQUE TRANSFER DEVICE
724 ! -----
725 ! Transfer of power from the engine to the transmission is specified with the
726 ! following parameters to specify a hydraulic torque converter or a mechanical
727 ! clutch.
728
729 INSTALL_TORQUE_TRANSFER_DEVICE ! VS Command to install a clutch or torque converter
730
731 OPT_CLUTCH          1 ! Torque transfer to transmission: 0 -> hydraulic torque
732 ! converter, 1 -> mechanical clutch, 2 -> torque converter
733 ! with lock-up clutch, 3 -> centrifugal clutch. The torque
734 ! converter options (0 and 2) use functions INV_CAP_TC and
735 ! RM_TC. Option 2 also uses LOCK_AT and UNLOCK_AT. [I]
736 OPT_CLUTCH_MODE     1 ! Clutch control mode: 0 -> open-loop with function
737 ! CLUTCH_CONTROL, 1 -> closed-loop (based on shifting and
738 ! function CLUTCH_TORQUE) [I]
739 OPT_CLUTCH_DELAY     1 ! Apply lag to requested clutch torque using time constants: 0
740 ! -> no, 1 -> yes
741 OPT_PWR_CPL_INTERNAL 1 ! Internal power coupling model (torque converter and
742 ! clutch): 1 -> internal, 0 -> external
743 AV_ENG_LOW_CLUTCH 400 ; rpm ! Engine speed to automatically disengage clutch at low
744 ! speed
745 TC_CLUTCH_DISENGAGE 0.001 ; s ! Time constant for clutch torque (disengage)
746 TC_CLUTCH_ENGAGE    0.1 ; s ! Time constant for clutch torque (engage)
747 T_CL_START          0.05 ; s ! Time elapsed after shift starts when clutch control
748 ! starts [I]
749 T_CL_DISENGAGE      0.15 ; s ! Time elapsed when clutch is fully disengaged [I]
750 T_CL_PRESS_TOTAL    0.5 ; s ! Time elapsed when clutch starts re-engaging [I]
751 T_CL_HALF_HOLD      0.3 ; s ! Time holding the clutch at half pressure [I]
752 T_CL_RE_ENGAGE      1.5 ; s ! Time elapsed when clutch is fully re-engaged [I]
753 ! T_CL_HALF_TOTAL    1 ; s ! CALC -- Time releasing the clutch during shift
754 ! T_CL_PRESS_HOLD    0.35 ; s ! CALC -- Time holding the clutch fully disengaged
755 T_GEAR_LAG          0.2 ; s ! Time elapsed after shift starts when shift is complete
756 T_TH_START          0 ; s ! Time elapsed after shift starts when throttle control
757 ! starts [I]
758 T_TH_ZERO           0.1 ; s ! Time elapsed when throttle drops to zero [I]
759 T_TH_ZERO_TOTAL     0.5 ; s ! Time elapsed when throttle starts to return [I]
760 T_TH_RETURNED        1.5 ; s ! Time elapsed when throttle has returned to pre-shift
761 ! level [I]
762 ! T_TH_ZERO_HOLD     0.4 ; s ! CALC -- Time spent holding zero throttle [I]
763 ! T_TH_RETURN        1 ; s ! CALC -- Time spent returning to original throttle [I]
764 VLOW_CLUTCH         1 ; km/h ! Disengage the clutch when effective wheel speeds are
765 ! below this limit when braking or resting [I]

```

Figure 56. Listing of closed-loop clutch parameters in the Echo file.



## Other Applications of the Closed-Loop Clutch Timeline

The timelines shown in Figure 55 (page 66) are applied as shown for upshift or downshift gear changes when the gear changes are separated in time by more than the time needed to perform a change. There are at least three conditions where the timelines are partially used.

### *Rapid gear changes*

Sometimes a simulation requires rapid gear changes, in which the time between shifts is shorter than the time specified for a full shift. For example, Figure 57 shows time histories for a vehicle with manual transmission coming to a stop at an intersection using the speed controller (lower-right plots). The vehicle is stopped at about 10s into the run. The upper-left plot shows the gear being changed sequentially from 6 to 1. The upper-right plot shows the clutch control during the shift changes and stopping of the vehicle.

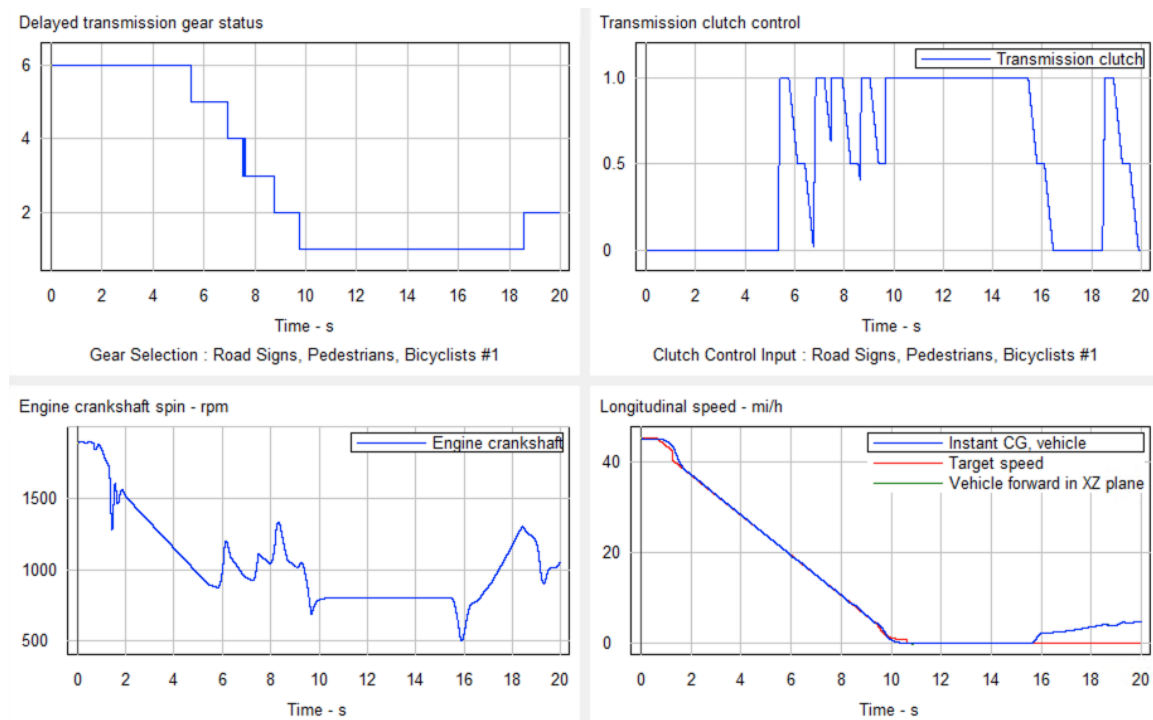


Figure 57. Clutch and gear time histories for a vehicle braking situation.

The first shift (gear 6 to 5) occurs at about 5.5s. Note the change in the clutch control, which shows the same characteristic shape as seen on the Browser screen (Figure 55). The next four shifts occur rapidly (between 7s and 9.6s), such that there is not time for the clutch control to follow the full timeline for each shift. The clutch control is partly disengaged from 7s to 16s. As shown in the plot, the clutch control for each new shift begins immediately without fully re-engaging the clutch. For example, at 7.5s, the shift from gear 4 to 3 is triggered. The clutch has not fully re-engaged from the previous shift (5 to 4), so it increases immediately starting from the current value.

### *Use of clutch to avoid engine stalling at low speeds*

The normal behavior of the closed-loop clutch controller is overridden at low speeds to avoid stalling the engine. When the forward vehicle speed drops below the specified limit `VLOW_CLUTCH` (15, Figure 55, page 66) and the lagged throttle is less than 0.01, the clutch will dis-engage if (1) the brakes are applied (indicating an intention to stop), or (2) a low engine RPM limit `AV_ENG_LOW_CLUTCH` is reached. In the example plots in Figure 57, the clutch is fully disengaged at 9.5s and does not begin to re-engage until 15.5s.

<b>Note</b> The parameter <code>AV_ENG_LOW_CLUTCH</code> is normally set on the <b>Powertrain Engine</b> screen. This is done to allow the same clutch settings to be used for powertrains using a variety of engines. If the powertrain does not have a clutch, the parameter is ignored and is not listed in the Echo file.
---

Several state variables define possible conditions of the clutch controller. Figure 58 shows part of an Echo file made at the end of a simulation, listing some of the state variables in the VS Math Model.

```
9965 SV_CL_ACCEL_START 0 ; - ! Boolean: low-speed accel mode started prev. time step
9966 SV_CL_AUTO_ACCEL 0 ; - ! Boolean: clutch is re-engaging to accelerate from stop
9967 SV_CL_AUTO_BRAKE 1 ; - ! Boolean: clutch is disengaging to stop
9968 SV_CL_AUTO_INIT 0 ; - ! Initiate closed-loop clutch, set in previous time step:
9969 ! -1 -> accel, 0 -> no change, or 1 -> brake
9970 SV_CL_AUTO_SHIFT 0 ; - ! Boolean: gear shift with clutch is in progress
9971 SV_CL_BRAKE_STOP 1 ; - ! Boolean: low-speed brake mode started prev. time step
9972 SV_CL_CON 1 ; - ! Transmission clutch control {ClutchTr}
```

*Figure 58. Several state variables define possible conditions of the clutch controller.*

State variables involving clutch control have keyword names that begin with `SC_CL_`. Some examples:

- `SV_CL_CON` is the current state of the clutch control (a value in the range 0 to 1).
- `SV_CL_AUTO_SHIFT` indicates that a gear shift is in progress.
- `SV_CL_AUTO_ACCEL` indicates that the clutch is disengaging to accelerate from stop.
- `SV_CL_AUTO_BRAKE` indicates that the clutch is disengaging to stop.

### *Use of clutch to accelerate from very low speeds*

The closed-loop clutch controller automatically re-engages at very low speeds if the lagged throttle is greater than 0.01. This mode is indicated by the state variable `SV_CL_AUTO_ACCEL` (Figure 58). This mode occurs in simulations where the speed controller brings the vehicle to a stop (or near stop) and then applies throttle to accelerate.

## **Using External Gear Shifting (Simulink, Driving Simulator, etc.)**

In some advanced applications, the shifting control input may be calculated by external code such as Simulink, External C, VS Commands, or hardware (e.g., driving simulator). The internal variables of the drive mode and gearshift position can be modified or replaced by the import

variables `IMP_MODE_TRANS` and `IMP_GEAR_TRANS`, respectively. The imported variables can be specified on the **I/O Channels: Import** screen. The selected gear position by the transmission is output via the variable `GearStat`.

If the simulated car uses closed-loop gear shifting, the drive mode can be commanded externally through the import variable `IMP_MODE_TRANS`. Therefore, to use only external commands, the internal value should be set to 0 (neutral). Table 10 summarizes external import/export variables for closed-loop gear shifting.

*Table 10. External importing/exporting variables for closed-loop gear shifting.*

<b>Gear Position</b>	<b>Drive mode</b> ( <code>IMP_MODE_TRANS</code> )	<b>Shift position</b> <code>IMP_GEAR_TRANS</code>	<b>Shift position output</b> ( <code>GearStat</code> )
Reverse	-1	0	-1
Neutral	0	0	0
Auto. shift 1 <sup>st</sup> – highest gear	<i>highest forward gear</i>	0	Gear selected by transmission

On the other hand, if the simulated car has open-loop gear shifting, the gearshift position can be commanded externally through the import variable `IMP_GEAR_TRANS`. In this case, the internally calculated gearshift position can be cancelled by selecting **External shift schedule** in the drop-down list in **Powertrain: Transmission (18 Gears or CVT)** screen (keyword = `OPT_SHIFT_INTERNAL 0`). Table 11 summarizes external import/export variables for open-loop shifting.

*Table 11. External importing/exporting variables for open-loop gear shifting.*

<b>Gear Position</b>	<b>Importing drive mode</b> ( <code>IMP_MODE_TRANS</code> )	<b>Importing shift position</b> <code>MP_GEAR_TRANS</code>	<b>Shift position output</b> ( <code>GearStat</code> )
Reverse	0	-1	-1
Neutral	0	0	0
Forward	0	<i>gear (1, 2, ... )</i>	<i>gear (1, 2, ... )</i>

## Appendix: Optimal Control Method for Steering Control

### Theory

The algorithm is intended to provide optimal control for a continuous linear system:

$$\dot{\mathbf{x}} = \mathbf{A} \mathbf{x} + \mathbf{B} \mathbf{u} + \mathbf{H} \mathbf{v} \quad (19)$$

$$\mathbf{y}_{\text{out}} = \mathbf{C} \mathbf{x} + \mathbf{D} \mathbf{u} + \mathbf{E} \mathbf{v} \quad (20)$$

where  $\mathbf{x}$  is an array of  $n$  state variables,  $\mathbf{u}$  is a control input,  $\mathbf{v}$  is a disturbance,  $\mathbf{y}_{\text{out}}$  is an output variable of interest, and  $\mathbf{A}$ ,  $\mathbf{B}$ ,  $\mathbf{C}$ ,  $\mathbf{D}$ ,  $\mathbf{E}$ , and  $\mathbf{H}$  are matrices with constant coefficients. The control objective is to determine the value of  $\mathbf{u}$  that causes the predicted output  $\mathbf{y}_{\text{out}}(t)$  to match a target

$y_{\text{targ}}(t)$  over some preview time  $T$ . In the case of vehicle steering control,  $u$  will be the steering and  $v$  will be rear steering due to suspension kinematics and compliance.

In the above equations,  $A$  is an  $n \times n$  matrix. In the general case,  $u$ ,  $v$ , and  $y$  could be arrays involving more than one control, disturbance and/or output variable. However, this derivation only considers the one case in which  $u$ ,  $v$ , and  $y$  are scalars.  $B$  and  $H$  are  $n \times 1$  matrices and  $C$  is a  $1 \times n$  matrix. There is a further simplification, namely, that  $y_{\text{out}}$  does not depend explicitly on  $u$  or  $v$ . Thus, the  $D$  and  $E$  matrices are not used.

Another simplifying assumption made for this analysis is that the control  $u$  and disturbance  $v$  remain constant over the preview time  $T$ .

If the system has initial conditions  $\mathbf{x}_0$  at time  $t = 0$ , a constant control input  $u$ , and a constant disturbance  $v$ , then the time response is:

$$\mathbf{x}(t) = e^{At} \mathbf{x}_0 + \int_0^t e^{A\eta} B u d\eta + \int_0^t e^{A\eta} H v d\eta \quad (21)$$

The term  $e^{At}$  is an  $n \times n$  matrix called the state transition matrix. Each coefficient in the matrix is the portion of state variable  $i$  at time  $t$  that is linearly related to state variable  $j$  at time 0, calculated by numerical integration. The product of the state transition matrix ( $e^{At}$ ) and the array of initial conditions ( $\mathbf{x}_0$ ) is an array of length  $n$  with the part of each state variable at time  $t$  due to the initial conditions of the system at  $t = 0$ . This is called the free response. The two integrals in Equation 21 define forced responses to each state variable due to the constant control  $u$  and disturbance  $v$  over the time interval.

Combining Equations 20 and 21 gives the response of the output variable  $y_{\text{out}}$ :

$$y_{\text{out}}(t) = C \mathbf{x} = C e^{At} \mathbf{x}_0 + C \left[ \int_0^t e^{A\eta} d\eta \right] [B u + H v] \quad (22)$$

A control-response scalar  $g$  is defined to relate the control input  $u$  over the interval  $t$  to the output variable  $y_{\text{out}}$  at time  $t$ .

$$g(t) \equiv \frac{\partial y(t)}{\partial u} \quad g(t) = C \left[ \int_0^t e^{A\eta} d\eta \right] B \quad (23)$$

A disturbance-response scalar  $h$  is defined to relate the disturbance  $v$  over the interval  $t$  to the output variable  $y$  at time  $t$ .

$$h(t) \equiv \frac{\partial y(t)}{\partial v} \quad h(t) = C \left[ \int_0^t e^{A\eta} d\eta \right] H \quad (24)$$

A free-response array  $F$  with  $n$  elements is defined to simplify the following notation.  $F$  relates the state variables at time 0 to the resulting output variable  $y$  at time  $t$ .

$$F(t) = C e^{At} \quad f_i(t) \equiv \frac{\partial y(t)}{\partial x_i(0)} \quad (25)$$

Note that  $g(t)$  and  $h(t)$  are related to  $F(t)$  by integration:

$$g(t) = \left[ \int_0^t e^{A\eta} d\eta \right] B \quad h(t) = \left[ \int_0^t e^{A\eta} d\eta \right] H \quad (26)$$

The response equation, re-written using the newly introduced terms, is:

$$y_{\text{out}}(t) = F(t) \mathbf{x}_0 + g(t) u + h(t) v \quad (27)$$

To determine the optimal control based on a preview time  $T$ , a quadratic performance index  $J$  is defined:

$$J = \frac{1}{T} \int_0^T [y_{\text{targ}}(t) - y_{\text{out}}(t)]^2 W(t) dt \quad (28)$$

where  $W(t)$  is an arbitrary weighting function. In this equation, time  $t$  is relative to the preview interval. In the controller view, the current time is always zero, and  $J$  is based on a prediction over the next  $T$  seconds.

A control  $u$  is considered optimal if it minimizes  $J$  — the squared deviation of the variable  $y_{\text{out}}(t)$  relative to the target function  $y_{\text{targ}}(t)$ . Because  $J$  is quadratic, the minimum occurs when the derivative  $\partial J / \partial u$  is zero. The value of  $u$  that minimizes  $J$  can be found by substituting Equation 27 into 28 and taking the partial derivative of  $J$  with respect to  $u$ :

$$J = \frac{1}{T} \int_0^T [y_{\text{targ}}(t) - F(t) \mathbf{x}_0 - g(t) u - h(t) v]^2 W(t) dt \quad (29)$$

$$\frac{\partial J}{\partial u} = 0 = \frac{2}{T} \int_0^T [y_{\text{targ}}(t) - F(t) \mathbf{x}_0 - g(t) u - h(t) v] g(t) W(t) dt \quad (30)$$

Solving for  $u$  gives the following:

$$u = \frac{\int_0^T [y_{\text{targ}}(t) - F(t) \mathbf{x}_0 - h(t) v] g(t) W(t) dt}{\int_0^T g(t)^2 W(t) dt} \quad (31)$$

In practice, the integrals over  $T$  can be replaced with finite summations for  $m$  intervals within the preview:

$$u = \frac{\sum_{i=1}^m [y_{\text{targ } i} - F_i \mathbf{x}_0 - h_i v] g_i W_i}{\sum_{i=1}^m g_i^2 W_i} \quad (32)$$

where the time dependencies of Equation 31 are replaced with an index  $i$ . The meaning here is that index  $i$  applied to  $F$ ,  $g$ ,  $h$ ,  $W$ , and  $y_{\text{targ}}$  refers to the value at time  $t = i T/m$ .

## Application

The solution involves a summation over  $m$  intervals ( $m$  is presently programmed as 10). For a given speed, the relation between interval  $i$  and time is fixed ( $t = i T/m$ ). Each time the steering controller is applied, it is provided  $\mathbf{x}_0$  (the current values of the four state variables) and the information needed to determine  $y_{\text{targ}}$  for each preview interval.

The free-response coefficients in the array  $F_i$  define the lateral position of the vehicle at the end of interval  $i$  due to non-zero initial conditions. Although the internal 2D vehicle model has four state variables, the choice of axis systems simplifies the calculations. Figure 15 showed that the initial values of  $x_1$  (lateral coordinate  $Y$ ) and  $x_2$  (yaw angle  $\psi$ ) are identically zero in the axis system of the steering controller. Therefore, only two coefficients are needed in array  $F_i$ :  $f_{1i}$  accounts for initial lateral velocity ( $x_3$ ), and  $f_{2i}$  accounts for initial yaw rate ( $x_4$ ).

The coefficient  $f_{1i}$  represents the value of  $y_{\text{out}}$  at the end of interval  $i$ , for an initial value of  $x_3 = 1$ . The values are calculated for  $i = 1, \dots, m$  using numerical integration. The initial values of all state variables except  $x_3$  are set to zero and  $x_3$  is set to unity. The 2D model is then simulated from  $t = 0$  to  $t = T$ , and values of the lateral position are saved at the  $m$  locations used in the summation.

The process is repeated to determine the values of  $f_{2i}$ , except that the initial conditions for the 2D model are that  $x_4 = 1$  and all other variables are 0.

The free-response coefficients in  $F_i$  are used with Equation 26 to compute the control response coefficients  $g_i$  and the disturbance coefficient  $h_i$ . The integral of  $y_{out}$  is calculated during the numerical integration by using a fifth state variable whose derivative is simply  $y_{out}$ .

Equation 32 is used to determine the steering control. The effect of front steer due to factors other than the driver is subtracted from this value to obtain the steer needed by the driver,  $u_c$  (see equation 12, page 21).

## A Linear 2D Vehicle Model

A simplified 2D vehicle model with forward speed ( $V_x$ ) and four state variables is used to predict the motion of the vehicle. The state variables are:

- $x_1$  = Local Y coordinate of the vehicle mass center, in the steering controller axis system,
- $x_2$  = Local yaw angle of vehicle, relative to the steering controller axis system,
- $x_3$  =  $V_y$ , the lateral component of mass center velocity in the vehicle axis system, and
- $x_4$  = Yaw rate.

The model is linearized by replacing cosine of local yaw with 1, and sine of local yaw with the local yaw. The A, B, and H matrices are then:

$$A = \begin{bmatrix} 0 & V_x & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{-(C_f + C_r)}{M |V_x|} & \frac{b C_r - a C_f}{M V_x} - |V_x| \\ 0 & 0 & \frac{b C_r - a C_f}{I_{zz} V_x} & \frac{-(a^2 C_f + b^2 C_r)}{I_{zz} |V_x|} \end{bmatrix} \quad (33)$$

$$B = \begin{bmatrix} 0 \\ 0 \\ \frac{C_f + C_r f(V_x)}{M} \\ \frac{a C_f - b C_r f(V_x)}{I_{zz}} \end{bmatrix} \quad H = \begin{bmatrix} 0 \\ 0 \\ \frac{C_r}{M} \\ \frac{-b C_r}{I_{zz}} \end{bmatrix} \quad (34)$$

where  $C_f$  and  $C_r$  are tire cornering stiffness coefficients for the front and rear axles,  $M$  is the total vehicle mass,  $a$  is the distance from the front axle to the mass center,  $b$  is the distance from the rear axle to the mass center,  $V_x$  is the forward vehicle speed,  $f(V_x)$  is the ratio of rear steer to front steer angle as function of the vehicle speed, and  $I_{zz}$  is the polar moment of inertia of the vehicle in yaw.

If the vehicle unit has more than two axles and the first two axles are part of a tandem suspension, then the parameter  $a$  is the distance from the mass center to the mid-point of the two front axles. If

the vehicle has more than one rear axle, then the parameter  $b$  goes from the mass center to the average of the distances to all rear axles.

The mass  $M$  used by the controller is obtained by summing the static axle loads. If the vehicle is towing a trailer, the hitch load will increase  $M$ .

The yaw inertia  $I_{zz}$  is calculated as  $a \cdot b \cdot M$ , so it will also have a larger value if a lead vehicle unit is towing a trailer.

The output variable of interest is the lateral position of the front axle. Thus, the C matrix is defined for linear behavior when the vehicle is moving forward as:

$$\mathbf{C} = [1 \ a \ 0 \ 0] \quad (35)$$

When driving backwards, the rear axle is used as the reference, and the C matrix is:

$$\mathbf{C} = [1 \ b \ 0 \ 0] \quad (36)$$

## Speed Sensitivity

The coefficients for free response  $F_i$ , control response  $g_i$ , and disturbance response  $h_i$  all depend on the forward speed  $V_X$  that appears in the A and B matrices shown in equations 33 and 34. The time step used for the numerical integration to calculate these coefficients must be small relative to the dynamics of the 2D vehicle model. The main factor in determining this time step is the magnitude of the coefficient  $a_{33}$  in Equation 33; for lower speeds, the time step is set to be inversely proportional to the magnitude of this coefficient:  $[C_f + C_r]/[M V_X]$

The steering controller code calculates values of the coefficients  $F_i$ ,  $g_i$ , and  $h_i$  for several speeds separated by about 1% of the current vehicle speed and uses linear interpolation to estimate their values at the current vehicle speed. This provides a continuous steering control without the need for running a computationally intensive numerical integration at every time step for the main vehicle simulation.

As the vehicle speed drops, the equations become numerically stiff, requiring a smaller time step for the internal numerical integration. Also, the system becomes less dynamical and more kinematical. As noted earlier, the model includes a parameter  $V_{LOW\_DM}$  to set a minimum speed for recalculating the coefficients. For speeds below this setting, the vehicle is controlled as if it were traveling at speed  $V_{LOW\_DM}$ .

## References

1. MacAdam, C.C. "Application of an Optimal Preview Control for Simulation of Closed-Loop Automobile Driving," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 11, June 1981.
2. MacAdam, C.C. "An Optimal Preview Control for Linear Systems," Journal of Dynamic Systems, Measurement, and Control, ASME, Vol. 102, No. 3, Sept. 1980.