755 Phoenix Drive, Ann Arbor MI, 48108, USA
Phone: 734 668-2930 • Fax: 734 668-2877 • Email: info@carsim.com
carsim.com

BikeSim 2019.1 New Features

This document lists the notable new features in BikeSim version 2019.1.

VS Solver: Architecture

VS Commands

User-defined functions

Users can now define new functions with optional arguments, local variables, and return value. The new functions are started with the `begin_function` command, and end with the `end_function` statement. These can process a series of equations.

New Boolean operators

New infix operators for Boolean operations (&, |, <, >, <=, >=, ==, ~=) have been added to VS commands so that conditional expressions can be more easily assembled. The existing routines `GT()`, `AND()`, etc. remain supported.

A new `IF`(*x*, *y*, *z*) "special function" was introduced for use in formulas. It differs from other functions by only evaluating two of the three arguments. The argument *x* is evaluated and if *x* not equal to *0*, the argument *y* is evaluated and returned as the value of the function; on the other hand, if *x* is equal to *0*, the argument *z* is evaluated and returned as the value of the function. This new function, along with the new User-defined function capability, allows for the conditional processing of different groups of expressions during a simulation.

Improvements for VS Events

The syntax for the `define_event` command has been simplified to have just two arguments, with the second being optional:

`DEFINE_EVENT` *formula [pathname]*

The reporting of Events in the Log file has been modified to more closely match the appearance of the Events as shown in the Event GUI screen. The *formula* argument can be a complicated Boolean statement, or just a variable name. It may even be a constant, e.g., 1, to indicate that the Event should unconditionally be triggered at the next time step.

The previous syntax for `DEFINE_EVENT` has been retained for a new command `MAKE_EVENT`:

`MAKE_EVENT` *variable operator reference [pathname]*

This provides backward compatibility for old datasets in which *reference* is a number that automatically converted from user-units of *variable* to internal units. It also provides support for the old `DELETE_EVENTS` command, which is still available.

Functions for moving object and ADAS sensor output variables

Output variables for moving objects have names that end with the number of the object, e.g., `S_Obj_3` is the station of object #3. Functions were added to allow the variables to be accessed using the object index, e.g., `S_OBJ(3)` returns the value of `S_Obj_3`. These functions support user-defined functions that are intended to be reused for multiple objects and sensor detections.

COM Interface

Four new COM functions have been added: `Run_Background`, `Run_dSPACE_Background`, `StopWindowsRun`, and `Set_RundSPACE` to `Run_dSPACE.exe`

`Run_Background` and `Run_dSPACE_Background` allow an end-user to run a simulation in the background, `StopWindowsRun` allows to stop the simulation, and `Set_RundSPACE` enables the possibility to pause, stop, and play after executing `Run_dSPACE.exe`.

Embedded Python

Python 3.6.5 (32-bit and 64-bit) has been embedded in the VS Solvers. New commands allow users to take advantage of Python to further access VS Table information, as well as some improved

debugging capabilities. New simulation examples have been developed to specifically help the user to work with and understand the VS embedded Python system. Embedded Python is now available for both Windows and Linux, but not RT systems.

Command Line Tools on Windows

We have provided two new executables, `VS_Solver_Wrapper_CLI.exe` and `vslm_commandline.exe`, to grant Windows users command line access to the solvers and license manager. These tools will be especially useful for setting up automation of simulations in larger systems.

`VS_Solver_Wrapper_CLI.exe` has new options to generate documents based on the file `Run_all.par` without running VS Browser nor running a full simulation with VS Solvers. The documents include the import variables, output variables, state variables, and `run_doc.par`. These are the same documents that the VS Browser can create and view from the run control screen. For more information type use the command -help.

Machine-generated Documentation

VS Solvers can generate documentation files based on the same data used to make a run. These include a list of state variables (text file), lists of output variables (text, tabbed text, or Excel), and lists of import variables (text, tabbed text, or Excel). The file is based on the vehicle configuration plus commands that add features (moving objects, sensors, reference points, etc.). The option has been added to generate an Echo file (Parsfile) without running a simulation. The Echo file documents all parameters, tables, VS Commands, and active imports. This can be handy for diagnosing problems involving complicated setups involving external software or target machines, as used with RT systems. Even though the system in use might not be capable of running the simulation, it can be used to generate an Echo file.

All state variables have keywords shown in End Parsfiles, which may be used by advanced users in VS Commands or to set custom initial conditions. They all begin with the prefix "`SV_`", e.g., `SV_XO` is the global X coordinate of the vehicle lead unit sprung mass coordinate system. Some are also output variables, e.g., the output variable `Xo` is the same as the state variable `SV_XO`. In these cases, advanced users may set the value of the state variable with the output variable name. In 2019.1, the documentation for state variables that are also installed as output variables includes the output name in braces {}, e.g.:

```
SV_XO (m) ODE: Global X coord. of sprung-mass origin {Xo}
```

Timing When Connecting to Simulink and Other External Software

When running under Simulink, imported variables were not available in the VS Solver at T=0 in versions prior to 2018.1. To avoid numerical instabilities, the VS S-Function did nothing at the first time step at T=0. The next step, all imports were defined, and communication was as expected. A side-effect of this behavior was that output written to the VS or ERD file began after the first step, rather than before the first time step. In version 2018.1, the initialization of VS Solvers was reworked such that Simulink imports were available. In the same release, the VS S-Function was changed to no longer skip calculation when T=0.

After the release of 2019.0, we received feedback that some existing Simulink models no longer functioned as expected. The most common reason was that some models had a Simulink "memory block" installed to avoid Simulink error messages about an algebraic loop. Those memory blocks cause the first set of import variables to be delayed, so the VS Solvers receive zeros for those import variables. To provide customers a method to use existing Simulink models made for older versions of CarSim, a new parameter was added. When OPT_SKIP_TSTART=1, the VS Solver replicates the old behavior by doing nothing when T= TSTART. (In Simulink, TSTART is always 0.)

VS Solver: Models

Closed-loop Speed Controller

The built-in speed controller (SC) was extended in several ways.

Acceleration control with OPT_SC 5

The built-in speed controller now has an acceleration-control mode, enabled with OPT_SC 5. This option uses proportional control to match vehicle acceleration Ax_Rd to an acceleration command Ax_SCcmd. This capability is of interest for some ADAS (automated braking) and automated driving products where desired vehicle pose is specified with acceleration control.

Easier conversion of old datasets with bad values of BK_SC_PERF

The speed controller has two stages of calculations. First, it determines a longitudinal acceleration that is requested to meet a target in speed or acceleration. Second, it generates throttle and/or braking controls to obtain the requested acceleration.

When preparing version 2018.1, a defect was resolved was found involving the internal usage of the parameter BK_PERF_SC. After the mistake was fixed, old values of BK_PERF_SC could be multiplied by G (9.80665 m/s^2) to obtain the same behavior. Given that this coefficient is an approximation, it turns out that many example simulations perform acceptably either way. However, others do require that the coefficient be converted. In these cases, a new option is provided by the parameter OPT_SC_2018. Set this parameter to 1 and SC will perform the multiplication internally. This may be helpful when performing version-to-version validation.

Motion Sensor Outputs

Measures of longitudinal and lateral *jerk* (the derivative of an acceleration) are now available when using a motion sensor which outputs accelerations. These are the new output variables of Jx_S*i*, and Jy_S*i*, respectively, where *i* is the sensor's identification number.

Moving Objects

Options for using moving objects to represent traffic vehicles have been extended. Mimicking some forms of vehicle behavior that required custom VS Commands or external control is now provided with built-in options.

When a moving object is linked to a reference path via the parameter PATH_ID_OBJ, a number of options are enabled for controlling the vehicle.

Direction for traffic vehicles

In past versions, object velocity had the sign convention of being positive when moving in the direction of the path (increasing station), and negative when moving back on the path (decreasing station). In 2019.1 the sign convention is changed to mimic vehicle speed: a direction parameter has been added to determine which direction the object is facing. The parameter for the ego vehicle is `OPT_DIRECTION`; for an object, it is `OPT_DIR_OBJ`.

Speed takes lateral position and path curvature into account

In past versions, object speed was defined as dS/dT where S is station along the path linked to the object. When there is no lateral coordinate, or when the path is straight, the path speed equals the object speed. However, when the path is curved and the object has a lateral coordinate, the object speed differs from dS/dT. In 2019.1 station is still obtained with an ordinary differential equation (ODE) based on speed `V_OBJ_`*o*, but the change in station is amplified or reduced based on L, path curvature, and the partial derivative of the lateral position with respect to station: $\partial L/\partial S$.

Acceleration for traffic vehicles

An output variable `A_Obj_`*o* was introduced in version 2019.0 for moving objects controlled through acceleration, as enabled by a new parameter `OPT_ACCEL_OBJ`. In this case, the acceleration is used as the derivative of the speed `V_Obj_`*o*, which is calculated with an ordinary differential equation (ODE) by numerically integrating `A_Obj_`*o*. In version 2019.1, the acceleration is also created and calculated for moving objects controlled by speed but not acceleration (that is, `OPT_ACCEL_OBJ` is zero, but `OPT_SPEED_OBJ` is not zero). In this case, the acceleration is calculated automatically by numerically differentiating `V_Obj_`*o*.

Brake lights for traffic vehicles

When moving objects are set up to use speed or acceleration to control forward motion, a new output variable `Bk_Obj_`*o* is created. The value is automatically set to unity if the deceleration is greater than a threshold specified with a new parameter (`AX_BRK_OBJ_ON`), or if the object is almost at rest (absolute speed is less than 0.1 m/s); otherwise, `Bk_Obj_`*o* is assigned a value of zero.

If the moving object is created using a dataset from the **ADAS Multiple Moving Objects** library and controlled via speed or acceleration, and the linked animator dataset is a dataset from the **Animator: Vehicles and Sensor Targets** library, and the animator dataset supports brake lights, then brake lights will be shown automatically when `Bk_Obj_`*o* is nonzero.

Polygonal shapes

Polygonal shapes for moving objects were introduced in version 2019.0. Because a conventional Configurable Function has an independent variable that must increase for each row, the X-Y or S-L coordinates were represented with two linked tables `POLY_SHAPE_XVAL` and `POLY_SHAPE_YVAL`. Those have been replaced in 2019.1 with a single table `POLY_SHAPE`.

VS Roads and VS Terrain

The maximum number of reference paths was increased from 200 to 500. The number of VS Roads was increased from 100 to 200. The number of supporting Configurable Functions used for friction, elevation, and boundaries were also doubled.

An alternative to a set of connected VS Roads was introduced in 2019.0: a single surface that can encompass many drivable areas called VS Terrain. Support for VS Terrain was completed in 2019.1 for Windows and Linux. This new version is more scalable and supports much larger scenes. VS Terrain is not supported on any real-time hardware-in-the-loop platforms for this release.

VS Terrain can be used for 3D data from external sources that were converted to the `vsterrain` file format, accessed with the VS Terrain interface. It is well suited for driving simulators and simulations involving high-quality visual rendering using graphics engines such as Epic's Unreal Engine. It is also used to support 3D tiles that can be used to assemble scenes with VS Scene Builder. The `vsterrain` file includes all information about elevation, slope, friction, and rolling resistance, all as functions of global X and Y coordinates.

When VS Terrain is used:

1. The solver loads a file in `vsterrain` format via the command `VS_TERRAIN_FILE`.
2. When this is done, all connections between model tires and moving objects are made using an internal VS Terrain interface.
3. The Echo file will not show any data related to VS Roads: no roads, no elevation Configurable Functions, no friction Configurable Functions, and no rolling resistance property.
4. The parameter `CURRENT_ROAD_ID` is set to 1 and is locked.
5. Any attempt to create a VS Road with the command `DEFINE_ROADS` will generate an error.

The `VS_TERRAIN_FILE` command cannot be used if a VS Road was added with the command `DEFINE_ROADS`. Attempts to do so will generate an error.

64-bit Support for VS/STI and TYDEX/STI

Support has been added for vehicles using VS/STI and TYDEX/STI-defined tires to work with 64-bit versions of MATLAB/Simulink. Two DLL files exist in the database directory `Extensions\User_Tire\vsStiSimpleTire_2018` — one for 32-bit, and another for 64-bit. If using 64-bit Simulink, the 64-bit `vsSTiSimpleTire.dll64` file will be referenced in the field for the VS STI module (DLL) on the **Tire (External)** screen.

VS Browser: Graphic User Interface (GUI)

Animator: Vehicles and Targets

When used to define a set of traffic vehicles that use speed or acceleration control and animation together with information from the **Animator: Vehicles and Targets** library, information is

automatically written in the All Parsfile for VS Visualizer to show brakes lights when the moving object is braking.

If a moving object is not controlled using speed or acceleration, then brake lights will not be visible unless other commands are created by the user.

Multiple Moving Objects

Speed control settings include direction

In support of more options built into using moving objects to mimic traffic vehicles, the objects now have a parameter `OPT_DIR_OBJ` that is used when the object is following a path. This has the same general purpose as the parameter `OPT_DIRECTION` that is used with the ego vehicle: when set to +1, the object travels along the path in the direction of increasing station; when set to –1, the object travels in the direction of decreasing station. Either way, the object speed is positive when the object is moving in the direction specified by `OPT_DIR_OBJ`.

In 2019.0, the drop down control for setting a method for controlling station and speed had nine options, with two covering negative speed. With the addition of the separate direction control, the number of options for controller station/speed has been reduced to seven.

Initial speed is set twice when speed control is used

When one of the three speed control options are chosen from the **ADAS: Multiple Moving Objects** screen, the speed of the moving object is updated every time step with a VS Command `EQ_OUT` that is written automatically in the Parsfile for the dataset. In version 2019.1, the specified speed is also written with a VS Command `EQ_INIT` such that the speed is set before the bulk of the calculations are made for the first time step. This is done to support caravans of vehicles, where the initial speed of a moving object is based on the initial speed of another moving object. For this to work, all speeds must be known during the initialization.

Miscellaneous

1. The labels and right-click notes for the **Animator: Reference Frame** screen were updated to include options for using formulas to control animation.
2. The three **Control: Speed (Closed Loop)** screens were updated to support a new option to automatically correct units of the parameter `BK_PERF_SC` taken from databased in versions 2018.0 and older.
3. The **Scene External Import** screen will show a text message if an imported `vsscene` file contains a VS Terrain file and/or animator assets. It also shows the pathname for the `vsterrain` file.
4. The **Road 3D Surface (All Properties)** screen has a simpler appearance if linked to a **Scene External Import** dataset for a VS Terrain file.
5. The **Find Text in the Database** dialog box maintains the history of the five previous search terms. The search history is cleared when the application is closed.

Other Tools: Stacks

VS Visualizer

VS Visualizer is the Windows tool used to view animations and plot from variables calculated by BikeSim. It is used for post-processing and for live animation with real-time systems and driving simulators. It is normally accessed from the BikeSim GUI using Plot and Video buttons, and may also be used independently to visualize results. Improvements for 2019.1 are:

- Loading times have been improved by over 100x on certain scenes with hundreds of thousands of reference frames. This speeds up all loading, although the effect is less noticeable on smaller scenes.
- Many more updates are emitted on the progress of file loading and processing. The previous version updated much less frequently, and for very large datasets the application could appear to be frozen.

VS Scene Builder

VS Scene Builder is a Windows tool used to create scenes involving surfaces, paths, and animator assets. The scenes are typically exported in a form that is then imported into BikeSim. VS Scene Builder is accessed by the BikeSim **Tools** menu. New for 2019.1:

- All tiles now have a complementary VS Terrain file. The VS Terrain was generated with the VS Terrain Utility. Now prop placement and moving animated assets will follow the true shape of the tiles. Tiles from VS Scene Builder with VS Terrain files are not currently support on real time, hardware in the loop, platforms.
- Improved dynamically scaling paths and graphics resulting in a cleaner view when zooming in.
- Additional settings to adjust generated VS Terrain and FBX files when importing OpenDRIVE.
- A new RoundaboutNetwork OpenDRIVE tile was added.
- New warning and error dialog popups when the VS Scene Builder detect formatting issues or defects in the OpenDRIVE file.

VS Terrain Utility

VS Terrain Utility is a new tool introduced in 2019.1 that provides a quick way to create VS Terrain files for use in Scene Builder tiles and other purposes. VS Terrain Utility is accessible through the BikeSim Tools menu.

VS Terrain Utility can convert FBX files into VS Terrain. During the conversion, you can modify the friction and rolling resistance based on the materials found in the FBX. There is also an optimization stop during conversion where duplicate vertices are welded together, which can significantly lower the memory footprint when overlap is common. A debug OBJ file can be output to assist in visualizing the internal terrain model in the VS Visualizer. The *VS Scene Tile Creation* memo describes how to use the VS Terrain Utility.

Documentation

The following Reference Manuals have been updated:

1. System Parameters in VS Solvers
2. VS API
3. VS Commands
4. VS Solver Programs

The following Screen documents have been updated:

5. ADAS Sensors and Target Objects
6. Animator Camera Setup
7. Animator Reference Frames
8. Animator Shapes and Groups
9. Animator Vehicles and Sensor Targets
10. Brake System
11. Paths and Road Surfaces
12. Plot Setup
13. Powertrain System
14. Procedures and Events
15. Rider Controls
16. Scene External Import
17. Suspension Systems
18. Steering System
19. Three-Wheeled Motorcycle Models
20. Vehicle (Motorcycle and Rider Bodies)
21. VS Scene Builder

The following tech memos have been updated:

22. GPS Coordinates (previously known as GPS Coordinate Outputs)

Database

New and Updated Examples

Improvements to table interpolation methods for example data

The chosen interpolation method for an input table should suit the data itself and its intended use. For example, if the table consists of just two points, it is hard to justify anything other than a line between them. Another example: a spline fit may be better than piece-wise linear for a table which will be differentiated. A comprehensive audit of input tables for BikeSim resulted in two interpolation-method changes in the following datasets:

- Control: Steering by the Closed-loop Rider Model; { Lane Change } Go to Right 1m, dS = 2m
- Control: Steering by the Closed-loop Rider Model; { Lane Change } Go to Left 1m, dS = 2m.

Changes to torque converter data interpolation methods are discussed in a subsequent section.

Torque converter data cleanup

The torque ratio tables associated with all torque converter datasets have been made consistent; they now all use the `LINEAR_FLAT` interpolation method, and the number of datapoints used has been reduced. Previously, the spacing of the datapoints was irregular, having relatively many around the coupling point and relatively few at/near stall.

The K-factor table associated with the 1800 cc engine's torque converter has been resampled, and the data has been extended past a speed ratio of unity to improve coastdown behavior. It now uses a `SPLINE_FLAT` interpolation method, consistent with other K-factor data in VehicleSim products.

Animator Assets

Brake lights

We updated all brake lights for all vehicles. When the brake lights are activated, we now display a noticeable glow effect around the lights. This makes the activated brake lights more noticeable when viewed from a long distance.

Brake lights on ego vehicles are activated by the output variable `Bk_Stat` being non-zero. Moving objects will also activate brake lights appropriately when vehicle movement is being controlled using speed or acceleration modes.