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

CarSim 2019.1 New Features

This document lists the notable new features in CarSim 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 `cs-lm-cli.exe`, to grant Windows users command line access to the solvers and license manager. These tools will be familiar to Linux users, as they are the same tools that come packaged with Linux installations. 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.)

Installation

The following additional distros are now supported:

- OpenSUSE 42 Leap
- SUSE Enterprise Linux Server 12

SUSE/OpenSUSE natively supports the CentOS/RedHat's RPM Package Manager format (.rpm). Users can install all OpenSUSE supported products using the documented installation packages and procedures that are for CentOS and RedHat distributions

VS Solver: Models

Built-In Electric Powertrain (EV)

A built-in electric powertrain model has been added. The model shares the same electric motor and battery with the hybrid model that was introduced in 2019.0. A drop-down control allows existing powertrain screens to be used to select one of three options: conventional internal-combustion engine, hybrid (engine + electric motor), and electric. By selecting either the hybrid or electric option, the existing torque converter and transmission links are replaced with links to the new libraries, **Powertrain: Hybrid/Electric System** and **Powertrain: Hybrid/Electric Power Management Control**. An EPA Closed-Loop speed control example and an Open-Loop throttle with regenerative brake example are included to demonstrate the behavior of the EV system.

Closed-loop Steering Controller

The built-in driver model (DM) for path following was extended in several ways.

Driving in reverse

The DM now supports path following while driving in reverse. The DM controller determines an intent to drive backwards by two possible means:

1. a powertrain model is installed, and the transmission mode (`Mode_Trans`) is -1 (valid for all powertrains, including hybrid (HEV) and electric (EV)), or
2. the closed-loop speed controller is engaged, and the target speed is negative.

In either of these cases, the preview point(s) for the driver model are located behind the rear axle, rather than leading the front axle.

In the process of adding the capability of driving in reverse, the existing controller equations were reviewed, and a few improvements were made. The technical memo documenting the controller was updated to provide more details about the math used. These improvements can lead to better performance in most cases. However, in case the previous behavior is needed for comparing new results with results from previous versions, a new parameter was added to specify that the older controller be used: if `OPT_DM_2019` is set to a value of 1, the equations from 2019.0 and earlier are used.

New single preview point controller

After some internal research, an additional controller option was added: when `OPT_DM` is set to 3, a single preview point is used. Rather than using a simplified linear dynamic model of the vehicle, the wheels are simply steered in the direction of the point. This method works well in limit conditions where the vehicle behavior is no longer linear. It handles “drifting” situations where large slip angles occur.

The command `INSTALL_DM_OUTPUTS` installs output variables for the global X, Y, and Z coordinates point(s). When this command is issued and `OPT_DM` = 3, the two import variables are added (`IMP_X_DM` and `IMP_Y_DM`), allowing the preview point to be imported from other software or defined with VS Commands.

Steer by torque

The DM controller now supports a steer-by-torque mode. Internally, it calculates a requested steering wheel angle. It then applies a torque to a torsional spring and damper to obtain the desired angle.

The vehicle steering system now includes steering stops for the front axle when steering by torque, either through the DM controller or with open-loop torque input.

Closed-loop Speed Controller

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

Support for hybrid and electric vehicles

The speed controller was extended to work with hybrid and electric powertrains.

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 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 acceptable 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 the speed controller will perform the multiplication internally. This may be helpful when performing version-to-version validation.

Improved Representation of Asymmetry

CarSim vehicles have supported asymmetry in terms of component properties (tires, springs, kinematics, etc.), sprung mass properties (including payloads), and suspension locations. In 2019.1 the support of vehicle asymmetry has been further improved.

Asymmetric unsprung masses

The unsprung mass for generic/independent suspension has been divided into three parts on each side of the vehicle:

1. Mass of the unsteered part of the suspension on one side.
2. Mass of the steered part of the suspension on one side.
3. Mass of the detachable tire/wheel assembly.

The spin inertia (Iyy) is split into two parts: (1) the detachable tire/wheel assembly, and (2) the spinning part of the suspension. The same is true for the Ixx and Izz moments of inertia.

In the case of a solid-axle suspension, the unsteered parts of both sides are combined into a single solid axle mass. Otherwise, it has the same parts.

Improved initialization

A number of calculated design and static loads were added to the Echo file, to help confirm that the load distribution is as expected, and to help diagnose occurrences where the loads do not match expectations.

The initialization calculations have been extended to take asymmetric loading into account more fully. A roll stiffness is calculated for each suspension, and used to distribute the resistance to asymmetric loading from payloads and the sprung mass.

Steering Intermediate Shaft Kinematics

Support has been added to represent the kinematic effects of a steering intermediate shaft via a new configurable function, ISHAFT_KIN. The function describes the rotation of the output end of the shaft as a function of the input rotation. When used with column assist power steering, the shaft is placed between the steering torsion bar and assist motor. When used with rack or gear assist, the shaft is between the steering column and the input to the torsion bar.

Steer Angle Stops

Steering stops are now supported for each wheel on the first axle. When steering is controlled by an input steer angle, the use of stops is inappropriate because the model can become stiff if steer stops are encountered and the steer command continues to increase. However, when controlling steering by torque input, the steer torque input can exceed the reaction torque and drive the system to unrealistic steer angles. This can also cause the model to fail. To prevent this, each wheel on the first axle (the only one controlled by torque) now has two parameters used to calculate a reaction torque when a specified steer angle is reached. The parameters are A_STR_STOP_(L or R) the maximum steer angle at each side, and K_STR_STOP_(L or R), the stiffness of a torsional spring providing resisting torque when the maximum angle is reached.

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.

Ego Vehicle Option

A new option has been added relating to the initialization of the ego vehicle. An issue was discovered in which the ego vehicle would appear with an incorrect elevation relative to its road path. This issue was fixed with new functionality that writes a corrected value to SV_STA_ROAD. In the event users want to disable this behavior, or in the case where specifying a value for SV_STA_ROAD manually is desired, this can be done using OPT_INIT_STA_ROAD. This new option defaults to an enabled state with a value of 1. However, if an end-user wants to set SV_STA_ROAD to their own value, use a Misc. yellow field to first set OPT_INIT_STA_ROAD to a value of 0. On the next line in the Misc. yellow field, set SV_STA_ROAD to the station value of choice.

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: ∂L/∂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 (RT HIL) 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, connections between the terrain, the vehicle's tires, and if used in the simulation, 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; this cannot be changed.
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

Brake lights for traffic vehicles

Moving Objects that represent traffic vehicles can be set up to have brake lights. This option is available when the moving objects use speed or acceleration control, and their animation shapes are specified using datasets from the **Animator: Vehicles and Targets** library. In this case, the data necessary to render the brake lights will be written into the `Run_all.par` for VS Visualizer.

If a moving object is not controlled using speed or acceleration, then brake lights will not be visible unless alternate means of controlling them are implemented by the user.

Reverse lights for ego vehicles

The screen **Animator: Vehicles and Targets** includes a link for an **Animator: Shape File** dataset to define reverse lights. The functionality is similar to the brake lights, but they will only appear for the ego vehicle when the transmission mode (`Mode_Trans`) is -1. This feature is not available for moving objects.

Multiple Moving Objects

Speed control settings include direction

In support of more options built into the moving objects to mimic the behavior of 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 **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.

Vehicle Screens

A checkbox for hitch information has been added to the two sprung mass screens for the motor vehicle: **Vehicle: Sprung Mass** and **Vehicle: Sprung Mass (from Whole Vehicle)**. If the checkbox is checked, these screens will show three coordinates for the hitch point, and also include a blue link for hitch properties. The linked hitch dataset has tables for hitch moments, plus a blue link for an animator shape for the part of the hitch attached to the leading vehicle.

When a simulation is run, the hitch will appear in VS Visualizer regardless of whether a trailer is also used.

Note that if the box is not checked, a hitch will not be installed on the ego vehicle and the hitch animation shape will not appear in the VS Visualizer. To support Backward Compatibility, the hitch can still be linked on the trailer's vehicle assembly screens, e.g., **Vehicle: Trailer with 1 Axle**.

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 **Control: Steering by the Closed-loop Driver Model** screen was updated to support a new option for using a single-preview point to control steering. It also has a checkbox to support old equations for the 10-point preview option.
3. The three **Control: Speed (Closed Loop)** screens were updated to support a new option to automatically correct units of the parameter `BK_PERF_SC` when databases from version 2018.0 and older are converted to 2019.1.
4. A checkbox "Show EV parameter only" was added on each of the two libraries **Powertrain: Hybrid/Electric System** and **Powertrain: Hybrid/Electric Power Management Control** to support the new electric powertrain option for electric vehicles (EV). An **Electric** option was also added to the ring-control selector on existing powertrain screens to allow to select EV system.
5. A newly introduced tire mass parameter `M_TIRE` may be obtained from an external tire model (e.g. MF-Tyre/MF-Swift, FTire, ext.). There is also an option to set the wheel mass manually on the **Tire (External)** screen.
6. The **Scene: External Import** screen will show a message if an imported `.vsscene` file contains a VS Terrain file and/or animator assets. It also shows the pathname to the `.vsterrain` file.
7. The **Road: 3D Surface (All Properties)** screen has a simpler appearance if linked to a **Scene: External Import** dataset for a VS Terrain file, due to the fact that VS Roads cannot be used in conjunction with VS Terrain.
8. 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 plots from variables calculated by CarSim. It is used for post-processing and for live animation with real-time systems and driving simulators. Normally accessed from the CarSim GUI using dedicated Plot and Video buttons, VS Visualizer may also be launched as a stand-alone application to launch `.vsrap` files.

Improvements for 2019.1 are:

- Loading times have been improved by over 100x on certain scenes with hundreds of thousands of reference frames. Loading of all scenes benefit from the load time improvements, although the effect is less noticeable on smaller scenes.

- Many more updates are presented during the progress of file loading and processing. Previously the updates were less frequent, 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 exported in a form that is then imported into CarSim. Access to this utility is accomplished via the CarSim **Tools** drop-down menu.

New for 2019.1:

- All tiles now have a complementary VS Terrain file, where the VS Terrain file was generated using the VS Terrain Utility (also available from the CarSim **Tools** drop-down menu). 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 supported on real-time hardware-in-the-loop (RT HIL) platforms.
- Improvements to dynamically scaling of paths and graphics which result 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 detects possible 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 CarSim **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 (**Help** > **Paths, Road Surfaces, and Scenes**).

VS Connect

VS Connect is a technology used to support co-simulation with two simulation environments and a VS Solver. For example, VS Connect is used to run CarSim with EPIC Unreal (for graphics and possibly terrain) and Simulink. It is currently used in two plug-ins: A Simulink S-Function and an Unreal plugin.

New for 2019.1:

Enhanced time synchronization with new Remote Clock Control (RCC) features

These features enable the VS Connect S-function for Simulink and the VehicleSim Dynamics Plugin for Unreal Engine to work together to tightly couple the simulation time of these two independent simulation environments.

Added update triggering

In addition to scheduled updates, where VS Connect automatically sends update data to connected peers on a pre-set schedule, the VS Connect API now has a function to allow an update to be triggered at any time.

Default values

Default values can now be specified for S-function output ports via the configuration file. This allows the user to specify the value that is output from the VS Connect S-function before the communication link is established and valid data is received from the remote simulation.

VehicleSim Dynamics Plugin for Unreal Engine

The CarSim plugin for EPIC Unreal, available from the Unreal market, has been replaced with an updated version that includes `carsim_64.dll`. The plugin works from within Unreal. If using the plugin, it is necessary to have a regular CarSim installation and associated licensing.

New for 2019.1:

- Improved time synchronization via VS Connect RCC with external simulation environments (e.g. Simulink).
- Unreal Engine Blueprints interfaces added which enable the user to create their own VS Connect enabled objects from scratch without the need for any C++ code.
- Enhanced remote manipulation of Unreal Actor objects via VS Connect. This enables Unreal Engine + Simulink co-simulation with the VehicleSim vehicle running inside of Simulink and a proxy object representing the vehicle within the Unreal simulation.

VS Software Development Kit (SDK)

The VS SDK (VehicleSim Software Development Kit) is a collection of APIs, tools, examples, and documentation to enable the development of new software and extensions to the core technologies of CarSim. Starting with 2019.1, it is available as a downloadable archive in ZIP or TAR format.

The SDK is intended for developers and advanced users wishing to integrate CarSim with custom code. In order to test any projects by running simulations, it is necessary to have the regular CarSim installation and associated license.

Solvers

Dynamically link libraries (.DLL) for CarSim for Windows, for both x86 and x64. Also included are Shared Objects (.SO) for x64 on Ubuntu, OpenSUSE, and CentOS Linux distributions.

Tools

- HostID Utility – Used for finding DeviceID to generate NodeLock Licenses
- LicenseManager – Windows and Linux versions of CarSim License Manager
- SolverWrapper – CommandLine Wrapper for the solvers, for x86 and x64 Windows and x64 Linux Platforms.

Documentation

Documentation outlining the use of the various Tools, APIs, and Solvers, as well as documentation outlining how to configure licenses.

Libraries and APIs

- VS API – Used for interacting with the CarSim and TruckSim solvers in a programmatic fashion. Comes bundled with a suite of examples for its use.
- VS Connect API – Used for connecting CarSim and TruckSim with other technologies through a common interface. Includes bundled example server and client projects.
- VS Output API – API for interacting with solver outputs from CarSim and TruckSim programmatically. Includes bundled example project.
- VS SharedBuffer API – Interface for interacting with VS Visualizer output.
- VS Table API – Used for manipulating tables within CarSim and TruckSim programmatically. Includes bundled Example project.
- VS Vehicle API – Used for initializing a vehicle programmatically.

Included alongside these libraries are a series of examples that showcase their use. Many of these examples were previously included within the database bundled with the software. These examples are no longer located in their previous locations, but instead within the VS SDK. Locations where example projects where removed are listed below. Mechanical Simulation recommends all users migrate to the VS SDK for work involving these examples:

- `Extensions\Custom_C`
- `Extensions\Custom_M`
- `Extensions\Custom_Py`
- `Extensions\Custom_VB`

Documentation

The following documents have been added to the **Help** menu:

1. Initialization in CarSim and TruckSim (Technical Memo)

The following tutorials have been updated:

2. CarSim Quick Start Guide

The following Reference Manuals have been updated:

3. System Parameters in VS Solvers
4. VS API
5. VS Commands
6. VS Solver Programs

The following Screen documents have been updated:

7. ADAS Sensors and Target Objects
8. Animator Camera Setup
9. Animator Reference Frames
10. Animator Shapes and Groups
11. Animator Vehicles and Sensor Targets
12. The CarSim and TruckSim Steer Controller
13. Driver Controls
14. Electronic Stability Controller (ESC)
15. Paths and Road Surfaces
16. Plot Setup
17. Procedures and Events
18. Scene External Import
19. Steering Controller (Driver Model Details) (removed: merged with Driver Controls)
20. Steering Systems
21. Suspension Systems
22. Tire Models
23. Twist Beam Suspensions: Using 2D Tables
24. Twist Beam Suspensions: Using VS Commands
25. Vehicles
26. VS Scene Builder

The following tech memos have been updated:

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

Database

Updated CPAR Archives

The CPAR used for the CarSim Quick Start Guide was replaced for 2019.1.

New and Updated Examples

Separate tire and suspension masses

The mass of the wheel/tire assembly may now be entered on a tire screen, rather than being included with the suspension's unsprung mass. This means various tires may be used with a single suspension kinematics dataset without having to recompute and reenter the unsprung mass values. Consequently, every suspension kinematics dataset, independent or solid axle, has been adjusted to account for the removal of tire mass. Additionally, every tire dataset has been assigned a mass value. Wheel/tire assembly spin inertias were also assessed and updated as part of this process.

Updated VS Commands for In-Wheel Motor examples

The VS Commands-based In-Wheel Motor examples have been updated to facilitate the vehicle traveling in reverse. This was done using the VS Command sign(x,y) and checking the sign of the output variable `ModeTran`. If `ModeTran` is negative, corresponding to the transmission in reverse, the sign of the calculated torques that are applied to the wheels is given a negative sign and the wheels will be driven in reverse, the same direction as the wheels driven by the internal powertrain. If `ModeTran` is positive, corresponding to the transmission in a forward gear, then the sign of the calculated torques is positive, and the examples behave as they have in previous releases.

New sprung mass and steering data for the 4WD compact pickup

The "Pickup, Compact: Part-Time 4WD (in 2WD)" vehicle assembly has been updated to better represent current trucks on the market. The increased length and mass of these trucks is represented by a new sprung mass dataset, while a new rack-and-pinion dataset replaces the recirculating ball system. Accordingly, the vehicle has been renamed as "Pickup, Midsize: Part-Time 4WD (in 2WD)". Minor adjustments were made to some of the other linked datasets (such as unsprung mass magnitudes, tire size, and aero reference length). The previous compact pickup data is still available in the database, and the "Pickup, Compact: RWD" vehicle is unchanged.

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 CarSim resulted in seven interpolation-method changes in the following datasets:

- Brakes: Cooling Coefficient; E-Class, Sedan: Rear
- Control: Steering by the Closed-loop Driver Model; { Vehicle Dynamics Tests } On-Center Steer Test
- Control: Steering Torque (Open Loop); Sine Sweep Torque

- Road: Off-Center Elevation Map, S-L Grid; { Ride } Cross-Slope Sine Sweep
- Steering: Power Assist Force; Pickup, Midsize
- Steering: Parking Torque for 2 Wheels; { E-Class } E-Class SUV
- Tire: Camber Thrust Coefficient; { Reference Tires } Typical Large Vehicle.

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

New powertrain for the compact utility truck

The compact utility truck previously appeared with a minimum powertrain or a 150 kW powertrain; the truck now appears only with a 32 kW, 5 speed-manual transmission powertrain, new for version 2019.1. Overall, the new configuration is more representative of vehicles in this segment.

Updated trailer hitches

Articulation limits for the fifth-wheel for pickup, ball, and pintle hitches have been updated to more-realistic values.

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.

Updates to two-axle flatbed trailers

- Modified: *2A Utility Flatbed Trailer* now uses the *Susp. Compliance: Utility Flatbed Trailer* dataset for both axles.
- Added: *2A Utility Flatbed Trailer (HD)* is a heavier variant with an animator shape for a pintle hitch. It is intended to model what is marketed as a "heavy duty" variant of a flatbed trailer.

Animator Assets

Brake lights

The animation assets for the brake lights have been updated for all vehicles such that they emit a distinct glow, making them more noticeable when viewed from a distance.

Brake lights on ego vehicles are activated when the output variable `Bk_Stat` is equal to 1. As noted earlier, moving objects will also have brake lights that activate when their movement is controlled using either speed or acceleration modes.

Reverse lights

Reverse lights have been added for all ego vehicles, activated when the transmission mode (`ModeTran`) is -1.