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

BikeSim 2023.1 New Features

This document lists notable new features in BikeSim version 2023.1.

VS Solver Architecture

Improvements were made in the overall architecture of the VS Solver libraries that are used to build and run VS Math Models in BikeSim.

Stages of Model Calculations

The architecture of the VS Solver has traditionally had two stages available for model extensions via VS Commands and custom wrappers connecting with application program interface (API) functions: kinematics and dynamics. Wrappers for external software tools such as Simulink dealt only with the whole model, sharing import and export variables once per time step.

The architecture was extended to organize model calculations into four specific stages each timestep:

1. **State :** the vehicle state (position and speed state variables) is known, and information about the environment is also updated (station, ADAS targets and sensors, wind, and path information for the driver model).

2. **Control :** the built-in rider controls are applied to provide steering and speed control variables.
3. **Kinematics :** variables are calculated that depend on the position and velocity information from the vehicle and controllers.
4. **Dynamics :** variables are calculated using the remaining equations in the model. ODEs for the entire VS Math Model are integrated to obtain the values available at the start of the next timestep.

The new organization was made to provide tighter connections with external software for controllers and replacements of vehicle components. It also improves options available for advanced users extending models with VS Commands and embedded Python.

> **Note** The four stages of model calculations were mostly supported in the 2023.0 releases of CarSim and TruckSim. They were partially in place for BikeSim 2023.0. They are fully supported in BikeSim 2023.1.

Import and Export Arrays for Multiple Wrappers

VS Solvers may be run under the control of wrapper programs in simulation environments that combine external model components with the VS Math Model, exchanging information using arrays of import and export variables. In past versions, communication with the simulation environment takes place once each timestep, such that the VS Math Model receives import variables before performing any calculations, and then copies output variables into an export array that is shared after all calculations in the VS Math Model have been completed.

As noted above, the calculations made in the VS Math Model each timestep are now organized into four stages, to support closer timing when extending models with external software. Features were added to support the use of multiple wrappers with a single VS Math Model:

- `OPT_MULTI_WRAPPERS` is a new user parameter available when using import/export arrays. It can be set to 0 (default) or 1. When set to 1, the VS Solver will connect with up to four wrappers connected for the simulation (one for each stage of the model calculations). Six new arrays are added (three for import and three for export) in this mode, for a total of eight arrays for exchanging information.
- `IWRAPPER` is a new hidden index parameter, managed in the same way as `IAXLE` that allows existing `IMPORT` and `EXPORT` commands to be used as needed to set exchange arrays for four wrappers.
- Each timestep, the VS Math Model makes up to four passes through the model calculations (each under the control of a different wrapper), performing only the calculations associated with the currently active wrapper.
- Machine-generated documentation files for Import variables now identify the stage in which the variable is used.
- Machine-generated documentation files for output variables now mention the stage in which the variable is calculated.

To support these new capabilities, four new Simulink S-Functions are provided in the release for Windows and non-RT Linux, as described in the VS Wrappers section (page 5).

The new process has no noticeable effect on the computation speed for the simulation.

New Mode for Import Arrays

VS Solvers may be run under the control of wrapper programs in simulation environments that combine external model components with the VS Math Model, exchanging information using arrays of import and export variables. The variables used to create the Import and Export arrays are defined before the simulation starts, using the `IMPORT` and `EXPORT` VS Commands. The `IMPORT` command identifies the import variable that will be added to the Import array and specifies a mode for usage within the VS Math Model.

In prior versions, three modes were available for variables associated with the Import array: `ADD`, `MULTIPLY`, and `REPLACE`. The mode could only be specified prior to the start of the simulation using the `IMPORT` command. After the run started, it could not be changed.

The mode of an Import variable may now be changed during the simulation using a new VS Command `SET_IMPORT_MODE`. In addition, a new mode `AVAILABLE` has been added. This mode indicates that the variable is in the Import array but is not currently being used by the VS Math Model. The intent is that the action of setting up the Import array has been separated from the actions of using the import variables, to allow complicated scenarios to be simulated in which external controller or model components are enabled and disabled for different time sections within the simulation.

VS Commands

Several new commands were added, and the effects of some existing commands were changed.

Adding Equations

New commands `EQ_STATE` and `EQ_KIN` were added to create new equations in more stages of the model calculations. Related commands were added: `DELETE_EQS_KIN`, `DELETE_EQS_STATE`, `RESET_EQS_KIN`, and `RESET_EQS_STATE`.

Equations added by the `EQ_IN` command now apply after the new State stage and before the new Control stage. Most existing examples using `EQ_IN` continue to work as originally intended.

PID Controllers

PID controllers can now be used in runs via the VS Command `PID_CONTROLLER()`. These controllers can be used with any symbolic math model variable to control the dynamic systems present in BikeSim. The controller will calculate the error between the input variable and a setpoint, and then output a control signal according to the size of that error, as well as its integral and derivative. See the example run *PID Controllers* > *DLC - Constant Target Speed w/ PID* and *Help* > *Reference Manuals* > *VS Commands* for more information.

Numerical Differentiation

Certain built-in variables in VS Math Models can be modified on the fly by the user via `IMPORT` statements. In certain cases, it is possible to “unlink” certain variables when using the `IMPORT` …

REPLACE command to overwrite the value calculated by the VS Solver. For example, the steering wheel angle STEER_SW is calculated by integrating the calculated steering wheel angular velocity StrAV_SW. If the steering wheel angle is replaced via an import statement, the steering wheel angular velocity is still calculated internally and no longer reflects the actual steering wheel velocity used in the solver, which is the derivative of the custom variable supplied by the user.

To remedy this, a new VS math command was added in 2023.1: DIFFERENTIATE(). When used with an EQ_... command, this new command takes in a symbolic variable and numerically calculates its derivative with a backwards-looking finite difference scheme, using up to 5 time steps.

> **Note** Calculating the derivative of discontinuous signals can produce extremely large instantaneous velocities. It's recommended to use MAX() and MIN() to filter out these very large values.

Modes for Import variables

The mode of an import variables may now be changed during the simulation using a new VS Command SET_IMPORT_MODE, typically using a VS Event. A new AVAILABLE mode can be set for import variables to add variables to the array shared with external software such as Simulink, without using the variable within the math model. The mode can later be changed as needed using SET_IMPORT_MODE.

Save_Div function

A new function SAFE_DIV is available for user-defined expressions to avoid divide-by-zero issues.

VS Math Models

Improvements were made to the features included in the BikeSim math models.

Hybrid/Electric Vehicles

The efficiency values of electric motors and generators used in hybrid and electric vehicles can now be imported and set directly via the import variables IMP_EFF_MOTOR and IMP_EFF_GNRTR, respectively.

A simulation involving an electric or hybrid vehicle can now be automatically terminated by setting the SOC_STOP parameter. When the vehicle's state of charge (battery level) drops below this value, the run terminates with a message written to the log file. This value should be a fraction of total battery capacity, and thus should be set between 0 and 1.

The previous versions did not provide import variables that allow an external model of the electrical system or battery. This version adds new import variables for the battery SOC (State of Charge), charging and discharging efficiencies and three other import variables which may need to import from the external model so users can implement their own battery model in an external tool (e.g. Simulink.)

The new import variables for the hybrid/electric powertrain are summarized in Table 1.

Table 1. Import variables added for the hybrid/electric powertrain.

Keyword	**Description**
IMP_EFF_MOTOR	Electric motor efficiency
IMP_EFF_GNRTR	Electric generator efficiency
IMP_SOC_BTTRY	Battery state of charge
IMP_PW_BTTRY_CHRG	Battery charge limit
IMP_PW_BTTRY_DIS	Battery discharge limit
IMP_VOC_BTTRY	Battery open-circuit voltage
IMP_V_BTTRY	Battery output voltage
IMP_A_BTTRY	Battery output current

Convert Tire Vertical Stiffness to a Configurable Function

The BikeSim tire model was extended to replace a linear tire vertical stiffness parameter (KT) with a 2D configurable function for the vertical tire force as a function of tire compression and tire inclination angle (FZ_TIRE). This was done to support the representation of tires with significant nonlinear relationships between compression and force. It also supports conditions where lateral stiffness is considered for conditions with significant motorcycle lean.

Convert Chain Stiffness and Damping to Configurable Functions

In the BikeSim chain drive powertrain model, the chain stiffness (K_CHAIN_STRETCH) and damping (D_CHAIN_STRETCH) parameters were replaced with configurable functions, F_CHAIN_X and F_CHAIN_VX, respectively. The GUI still writes linear coefficients for those configurable functions in the parsfile; however, they can support tabular form through the use of the generic table screen.

VS Wrappers

Multiple Simulink S-function Blocks

Four versions of the VS S-Function have been made that connect with the VS Math Model during different the stages of the calculations made each timestep that were described earlier (page 2): vs_state (State), vs_ctl (Control), vs_kin (Kinematics), and vs_dyn (Dynamics). The new S-Functions were added to the VS library that is accessed from Simulink. These VS S-Functions provide multiple connection points between the VS Math Model and Simulink that are applied in series each timestep by Simulink.

Any combination of two to four of the new S-Functions may be used. The timing in Simulink is such that the last S-Function is applied at the end of the timestep, and the others are applied earlier, in sequence with other blocks in the Simulink model. The result is that replacement of parts of the vehicle model (tires, springs, brake controllers, etc.) is handled inline, without the delay of waiting until the next timestep.

Examples have been added to illustrate the use of the new serial S-Functions, and a technical memo has been added to describe the operation.

> **Note** The new S-Functions were included in BikeSim 2023.0 but were not supported by the VS Solvers. The VS Solver Architecture was extended in 2023.1, and now supports the multiple S-Functions.

Improved Support for Existing S-function Blocks

Past versions of BikeSim included a feature to use Kinematical Preview to avoid time lags for Simulink models that calculate forces and moments that can be imported into the vehicle model using kinematical information that was exported from the vehicle model. This option is set using a checkbox on the **Models: Simulink** screen **Sync kinematical exports with force imports**. That checkbox in turn generates text to set the parameter `OPT_IO_SYNC_FM` to 0 or 1. The option works for kinematical variables that are calculated for export early in the timestep. As part of the reorganization of the calculations done for this release, the Sync feature is supported for more components that calculate forces and moments based in kinematical information.

Timeline for Starting a Simulation

A Simulation is started by reading input files that define the layout of the model and values for parameters in the model, and then initializing the model. At a certain point, the model is "locked" in the sense that the number of ordinary differential equations (ODEs) is set. In recent versions, that point occurred at the end of the process of reading input files. When running under external control via the VS API, this step was performed by the API function `vs_setdef_and_read`.

The locking of the model now takes place slightly later, in the step performed by the API function `vs_initialize`. This allows custom wrapper programs to extend the model via API functions after input Parsfiles have been read, but before the model is initialized. For example, this change allows a wrapper to install the speed controller in support of other options used by the wrapper.

VS Browser: Graphic User Interface (GUI)

Minor changes were made to some of the existing screens.

1. The **Models: Simulink** screen has a new drop-down control for models with three kinds of VS S-Functions:
    a. The Simulink model has a single VS S-Function for the simulation.
    b. The Simulink model has multiple VS S-Functions, each for a separate vehicle. The vehicle simulations are run in parallel in Simulink using the **Tools > Parallel VS Math Models** screen.
    c. The Simulink model has multiple VS S-Functions, each for a different stage of the VS Math Model calculations. The S-Functions are run in series to support in-line extensions to the VS Math Model from Simulink. When this mode is selected, the screen blue links to four sets of I/O ports, each for a different stage and associated S-Function.

Note This change was made in 2023.0, but the corresponding updates in the VS Math Models were not completed for that release. The VS Solver Architecture was extended in 2023.1, as described earlier. The changes in the VS Browser now support those new capabilities.

2. The **Tire** screen has a new drop-down control for defining the tire vertical stiffness with two different ways:
    a. A linear spring rate (keyword = `FZ_TIRE_COEFFICIENT`) or
    b. Vertical force is a 2D configurable function of the vertical deflection and inclination angle.
3. The **I/O Channels Import** screen now supports the selection of the `AVAILABLE` mode for import variables specified on the screen.

VS Software Development Kit (SDK)

The VS API has been refined to support more interactions between wrapper programs and VS Math Models. In the 2023.0 and 2023.1 versions, the calculations done each timestep have been reorganized to perform model calculations in four specific stages each timestep:

1. **State :** the model state (position and speed state variables) is known, and information about the environment is updated.
2. **Control :** built-in driver controls are applied.
3. **Kinematics :** variables are calculated that depend on position and velocity information.
4. **Dynamics :** variables are calculated using the remaining equations in the model, including forces, moments, accelerations, and outputs that depend on these variables.

Support for Multiple Wrappers

The four stages of the calculations can each be associated with a separate wrapper program, with the possibility of supporting up to four wrappers that can work closely with the VS Math Model.

Support for multiple wrappers requires the following:

1. The wrappers must run in a sequence matching the stages of the VS Math Model equations.
2. The wrappers must each link to the same VS Solver library such that there is a single VS Math Model running.
3. The wrappers and VS Solver must all run on the same CPU core.

This capability is now supported for Simulink users with the inclusions of four VS S-Functions, each associated with a different stage.

VS API Changes

Defining Import and Output variables

Import and output variables are defined in a VS Solver with API functions such as `vs_define_imp` and `vs_define_out`. These create an internal symbolic structure with an attached number (double), units, description, and other properties. New API functions `vs_define_imp_where` and `vs_define_out_where` were added for version 2023.0 to include information about where the import is used, or the output is calculated. In the current version (2023.1) all import and output variables in the VS Math Model have the stage identified. This information is obtained from machined-generated documents provided in simple text or csv spreadsheet format, viewed by screens in the VS Browser that are used to select import or output variables.

Calling the vs_statement API function

The `vs_statement` API function allows a wrapper program to apply VS Commands programmatically, rather than by reading from an input Parsfile.

As noted earlier, the locking of the VS Math Model now takes place a little bit later in the startup, such that API function calls can be made after `vs_setdef_and_read` but before `vs_initialize`. This allows more state variables to be added, including ODEs.

Callback locations

The VS Solver has several functions to install callback functions that can be deployed in multiple points in the simulation timeline. The main one for calculations is the calc callback, installed with the API function `vs_install_calc_functions`. There are now 12 locations each time step where callback functions can be applied, including the four stages that now exist each timestep.

Examples

A new SDK example was added to show how the speed controller can be installed programmatically. This was not possible in past versions, because the controller installed an ODE state variable needed for integral control. With the re-ordering of the simulation setup, this step can occur after the input files have been read but before the model initialization is performed.

Documentation

The following document was added to the **Help** menu:

1. Technical Memos > VS Support for Multiple S-Functions

The following Guides and Tutorials were updated:

2. Running a VS Math Model in Simulink

The following Reference Manuals have been updated:

3. VS Browser (GUI and Database)
4. VS COM Interface

5. VS Commands
6. VS Commands Summary
7. VS Math Models

The following Screen documents have been updated:

8. Model Extensions and RT > External Models and RT Systems
9. Model Extensions and RT > Import and Export Variables
10. Powertrain > Powertrain System
11. Run Control Screen (Home)
12. Tire Models
13. Tools > Running with Parallel Solvers

The following Technical Memos have been updated:

14. Example: Extending a Model with VS Commands and the API
15. Example: Multiple Ports in Simulink for Sensors
16. Numerical Integration in VS Math Models
17. VS Solver CLI Wrapper
18. vs_sf VS Connect Server

The following Real-Time and DS System documents have been updated:

19. dSPACE RT Guide

The following SDK documents have been updated:

20. The VehicleSim API

Database

Additions were made in some of the run categories. The following new categories (with associated CPAR archive files) were added.

Batch Iteration > Batch Iteration Simulink, was created to use a matlab script and Simulink model to iterate a run-time variable [SPEED_TARGET_CONSTANT], for a batch of runs.

External Battery Model > External Battery, was created to demonstrate the use of the new import variables for external electrical system and battery specified in Simulink.

PID Controllers > DLC, Constant Target Speed w/ PID, was created to demonstrate the use of the new generic PID controllers.

Several *Simulink Multiple S-Functions* examples were made using the new four S-Functions for different stages of the model.

Tire Vertical Force w/ 2D Table > Tire stiffness as a function of inclination, was created to demonstrate the use of 2D table to define the tire stiffness that adds one more independent variable of tire inclination in order to include the effect of the tire's lateral stiffness.