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

BikeSim New Features

This document lists notable new features in BikeSim 2024.2 and other releases going back to 2022.0.

BikeSim 2024.2 New Features

VS Math Models

VS STI Callback Functions from VS Solvers

Various VS Solvers functions are available for use in the VS STI module as callbacks. In the past, any tire model using VS STI module required all callback functions to be defined. In this release, all these functions are optional for a tire model, i.e., a tire model needs to define them only if they are used in the specific model.

VS Embedded Python

The Embedded Python versions of VS API functions to get and set values for variables in the VS Math Model worked with most types of variables that have keyword names, such as parameters, state variables, output variables, and miscellaneous named variables. The Embedded Python versions now also work with import variables (those with keywords of the form `IMP_`*name*).

External Table Files

About half of the calculations made in VS Math Models involve tables that are used to calculate a dependent variable from one or two known variables. In most cases, tabular data from test rigs or other software are routinely put into the VS Browser via copy/paste. However, there are occasions where simple copy/paste into VS Browser table screens is inefficient. Alternative methods are available to provide tables with three types of external data files:

1. Parsfiles added to the model with an `INCLUDE` statement.
2. Binary VSTB files made from a VS Math Model or a VS utility program.
3. Output VS or ERD files.

These capabilities are supported with new examples, new documentation, and improvements in the VSTB file option. A new Tech Memo describes some of the examples and documents a new utility `vs_csv_to_vstb` that creates VSTB files from spreadsheet CSV files.

Regulating Hybrid/Electric Powertrain Motor Torque under Low Speed

New parameter (`VLOW_POWER_TRQ`) has been added in order to regulate motor demand torque at low speed. The motor demand torque used to be calculated by the power demand from driver divided by either transmission output speed or a magic number of 10 rad/sec whichever the larger value. With such calculation, the motor demand torque could be very large value when the vehicle speed is very low. In order to regulate the large demand torque at low speed, the magic number is replaced by the transmission speed derived from the new parameter. Setting `-1` to the new parameter provides the same behavior as prior versions for backward compatibility.

VS Terrain

The VS Terrain Utility and Library introduces version 3 of the format, which has been reworked to build terrain files much faster, while handling much larger scenes. The api allows for geometry to be added in bulk, in a format used in the most common file types. There is also a new V3 Virtual build type that can position many terrain files within a virtual space using the new .vsatlas file format.

Documentation

The following documents were added:

1. Reference Manuals > VS Embedded Python Interface
2. Technical Memos > Options for Loading Table Data into VS Math Models
3. Technical Memos > VS Embedded Python Interface Examples

The following VS Reference Manuals were updated:

4. System Parameters in VS Math Models
5. VS Commands
6. VS Math Models
7. VS Output API
8. VS Visualizer

The following documents in the **Help** menu were updated:

9. Model Extensions and RT > Custom Forces and Motion Sensors
10. Paths, Road Surfaces, and Scenes > Paths and Road Surfaces
11. Paths, Road Surfaces, and Scenes > VS Terrain
12. Powertrain > Electric and Hybrid-Electric System (BEV/HEV)

The following Technical Memos were updated:

13. Connecting to External Tire Models with VS STI

The following RT and DS documents were updated:

14. A&D RT Guide
15. Concurrent RT Guide
16. NI RT Guide
17. Speedgoat Guide for VehicleSim Products

The following VS SDK documents were updated:

18. Extending a Model with VS Commands and the API

Database

Examples were added or updated in the following categories.

Embedded Python

Nearly all existing examples for Embedded Python were replaced, and new examples were added to the same category. The category has a '*' prefix for this release.

External Table Files

Example runs show use of three kinds of external table files: Parsfiles loaded with the INCLUDE statement, binary VSTB files made with the VS Math Model or the new vs_csv_to_vstb tool, and tables made with data from output VS or ERD files.

SDK: Extended Models

The example path following steering controller was updated to remove some complexity dealing with initialization that was needed for older versions of the models (prior to 2023). The examples are now simpler and avoid a minor discontinuity at the start.

Real-Time

Speedgoat

Running VehicleSim FMU2.0 is now supported on Speedgoat target.

NI VeriStand LinuxRT

Support Simulink S-Function "vs_sf" running on NI VeriStand Linux RT system.

ETAS

Running VehicleSim on ETAS target is no longer supported.

BikeSim 2024.0 New Features

VS Solver Architecture: Delayed Initialization

A new option to delay initialization was added to allow imported variables from external software such as Simulink to be used to initialize the vehicle state before the simulation starts. This is needed when variables that are critical for determining the initial state of the model are provided by user VS Commands or imported from external software. A new system parameter `OPT_DELAY_INIT` is used to choose an alternate time to perform the initialization. Details are provided in a new Tech Memo, *Initialize a Vehicle with Imported Ground Z*, and new examples were added to the database.

VS Math Models

3rd-Party Tire Model on Linux

Siemens MF-Tyre/MF-Swift tire model is now supported on Linux 64-bit system environment.

Real-Time Platform: Vector CANoe

The VS Solvers now support the Vector CANoe Real-Time platform, which includes VN8900, VT6000, and RT Rack. These devices together form the Vector Tool Platform and are equipped with a Windows operating system. The CANoe platform is supported by the existing VS Browser (using the **Models: Transfer to Remote RT Target** library). Example runs were added to the database and the **Real-Time and DS Systems > Vector CANoe Guide** documentation was added.

Documentation

Built-in Help Documents

The following Technical Memo was added:

19. Initialize a Vehicle with Imported Ground Z

The following Reference Manuals were updated:

20. System Parameters in VS Math Models
21. VS Browser Reference Manual
22. VS Commands Reference Manual
23. VS Math Model Manual

The following RT and DS documents were updated:

24. A&D Guide
25. Concurrent RT Guide
26. dSPACE RT Guide
27. ETAS RTPC Guide
28. NI RT Target Systems

29. RT-Lab Target Systems
30. Speedgoat Guide

Japanese Language

Japanese language documentation is now available from the user section of the Mechanical Simulation website. The documents available at the time of this writing are translated from versions 2023.1 and 2023.2, with 2024.0 equivalents available later.

Database

New examples were added in the following category.

Imported Ground Elevation

New simulations are added to show several methods for handling user-defined road surfaces using Simulink or VS Commands. The examples make use of the new system parameter `OPT_DELAY_INIT` that allows imported variable to be used during the initialization of the vehicle model. Details are provided in a new Tech Memo, *Initialize a Vehicle with Imported Ground Z.*

BikeSim 2023.1 New Features

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:

31. **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).
32. **Control:** the built-in rider controls are applied to provide steering and speed control variables.
33. **Kinematics:** variables are calculated that depend on the position and velocity information from the vehicle and controllers.
34. **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 9).

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 6): `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 Tutorial was updated:

2. Running a VS Math Model in Simulink

The following Reference Manuals were 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 were 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 were 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 document was updated:

19. dSPACE RT Guide

The following SDK document was 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 to include the effect of the tire's lateral stiffness.

BikeSim 2023.0 New Features

VS Solver Architecture

VS Commands

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

1. The `DEFINE_OUTPUT` command was improved to work more effectively. `SET_OUTPUT_SHORT_NAME` was added to allow the manual creation of short (ERD compatible) output names by the user.
2. The `RETURN` command used in defined functions was improved to reduce problems. `DEFINE_LOCAL` was improved to eliminate option of indexed local variables.
3. Error checking was added for lengths of new variables and parameters. The name lengths were extended to 48 characters; longer names now generate error messages.

Echo Files

The display of information in Echo files generated for each simulation has been extended to indicate which parameters or table properties have been assigned values by reading from file, as opposed to those that keep default values. Some changes in this release are:

1. The convention for identifying a parameter whose value was not specified by any of the input files is to prefix the description with `[D]` (default). This has been extended to include properties of Configurable Functions.
2. A parameter `OPT_ECHO_DEFAULT` can disable the display of the `[D]` prefix, which is sometimes helpful when monitoring differences in Echo files. Another option has been added, in which the Echo file shows only parameters and Configurable Function properties that were specified in input Parsfiles. These files are much shorter and can be convenient for some applications involving automation and large numbers of runs.

Other Improvements

1. All error messages were reviewed; many were updated to provide more specific information about the cause of the error.
2. New VS API functions were added to provide a location in the equations of the model for import and output variables; these are described in the API Reference Manual in the VS software development kit (SDK). C examples in the SDK demonstrate the new functions.

VS Math Models

VS Tire Tester

For external tires, VS Tire Tester iteratively solves for the wheel center height needed to produce the requested tire vertical force. Beginning in version 2023.0, the maximum number of allowed iterations, as well as the solution tolerance, may be adjusted with the math model keywords

`TT_MAX_ITERATIONS` and `TT_TOLERANCE_FZ`, respectively. The default values of 1000 iterations and 1.0 N are consistent with the calculations of previous versions.

Advancing the Interface to External Tire Models

In past versions, VS Math Models connected to external tire models one tire at a time to get forces and moments of each tire. Some third-party tire models support parallelized calculation of all tires. The VS Solver and external tire model interface have been extended for the parallelized calculation of the external tire model namely COSIN FTire. The new interface speeds up the simulation by about a factor of three compared with the old interface for FTire.

The external tire model interface to COSIN FTire also adopts a dynamic library loader that automatically detects the FTire solver as installed by the FTire installer.

Electrified powertrain enhancement

Back-spinning of wheel with electric motor is prevented when the battery is recharged with regenerative brake.

VS Wrappers

dSPACE RT Unreal Live Animation and Remote Data Access

The BikeSim solver wrapper for dSPACE SCALEXIO 64-bit Linux has been enhanced to utilize VS Connect to provide remote data access and synchronization capabilities.

This feature can be used to produce Live Animation of RT simulation runs using the VehicleSim Dynamics plugin for Unreal Engine. The VS Connect features of this wrapper also allow advanced users to use the VS Connect API to write custom C/C++ programs that remotely access Imports and Outputs of the BikeSim solver while it is running on the RT system.

For more information, see the example Run **{RT: dSPACE} SCALEXIO: Unreal Engine Live Animation**", and the new tech memo **Unreal Engine: Live Animation with Simulink**.

Support for Speedgoat Real-Time Platform

The VS solvers now support the Speedgoat Real-Time platform: Performance and Mobile. The RT system is QNX 7.1 64bit. Software is MATLAB/Speedgoat R2001a or newer. The Speedgoat platform is supported by the existing VS Browser (using the **Models: Transfer to Remote RT Target** library). Two example runs were added to the database and the **Model Extensions and RT > External Models and RT Systems** documentation was updated.

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 is in support for an extension of the architecture of the VS Math Model. That change was not ready for the 2023.0 release of BikeSim, but is expected for the 2023.1 release.

2. The **Payload: Custom** screen has a new drop-down control to specify the reference for the Z coordinate of the payload: sprung mass coordinate system or trailer front hitch height. (This option is like one that was added in 2022.1 to the **Payload: Box Shape** screen.)
3. External tire interface to COSIN FTire model adopts a dynamic library loader which automatically detects FTire solver under FTire installation. Therefore, **Tire (External)** screen has changed to remove the **Tire program file (DLL)** field when FTire model is selected.

VS Software Development Kit (SDK)

The `vs_output` API has been updated to allow `.mat` file I/O.

Documentation

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

1. Real-Time and DS Systems > Speedgoat Guide for VehicleSim Products

The following Reference Manuals were updated:

2. System Parameters in VS Math Models
3. VS Browser (GUI and Database)
4. VS Commands
5. VS Commands Summary
6. VS Math Models

The following Screen documents were updated:

7. Animator > Vehicles and Sensor Targets
8. Model Extensions and RT > External Models and RT Systems
9. Paths and Road Surfaces
10. Payloads

11. Setting Up Import and Output Variables
12. Tires > Tire Models (Motorcycle)

The following Technical Memos were updated:

13. Example: Extending a Model with VS Commands and the API
14. GPS and UTM Coordinates

The following Real-Time and DS System documents were updated:

15. dSPACE RT Guide
16. VI-Grade DriveSim & VehicleSim Dynamics

The following SDK documents were updated:

17. The VehicleSim API
18. VS Output API: Reading and Accessing VS Output Files

The following Unreal documents were updated:

19. Unreal Engine & Python Co-simulation using VS Connect
20. VehicleSim Dynamics plugin for Unreal Engine example using VS Connect

Database

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

Alternative Coordinates

VS Math Models can now generate Universal Transverse Mercator coordinates, as shown in a new example run.

Echo File Options

The Quick Start Guide example was run with different settings that affect the information shown in Echo files.

New Signs

Several runs were made showing many highways signs (VS Visualizer resources) that were added to the database.

New VS Commands

An improved steer controller example was added.

BikeSim 2022.1 New Features

VS Solver: Architecture

VS Commands

1. The `PARTIAL` and `PARTIAL2` VS Commands have been added to return the partial derivative of a Configurable Function in either the row or column direction. The new `INVERSE` function returns the inverse of a Configurable Function if it is available.
2. `INSTALL_DM_IMPORTS` has been added to allow for installing Driver Model imports only. Previously, the only way to install them was to use `INSTALL_DM_OUTPUTS`, which also (still) installs them.

Other Improvements

1. The embedded Python included with all products has been updated to 3.10.2.
2. The user can now specify pre-processing and post-processing callback functions to be before and after the solver executes. These are available via the VS API or the GUI.
3. Support dSPACE SCALEXIO Linux 64-bit real-time system (dSPACE RLS 2022A and newer)
4. Support OPAL-RT RT-Lab Linux 64-bit real-time system.
5. The solver is now working with 32-bit and 64-bit LabVIEW.

VS Math Models

Powertrain Improvements

Improvements were made in the automated shifting behavior in the powertrain, which affects the powertrain behavior and the options for setting up driver control options. Most of the improvements involve the closed-loop clutch controller.

1. The location of parameters in the Echo file was adjusted to better link parameters to parts of the overall powertrain model. For example, throttle delay was moved from the Engine section (that only exists for powertrains with an internal combustion engine) to the overall powertrain system section.
2. Parameters `T_CL_START` and `T_TH_START` were added to provide more options for setting the timing for automated clutch and throttle modulation during shifting.
3. The calculations of clutch, throttle, and shift were improved to handle new requests for clutch/throttle activity if an activity was already in progress. For example, if a new shift is request before a shift in progress has ended, the new shift is started, with the start time adjusted automatically to avoid discontinuous jumps in clutch.
4. The closed-loop clutch behavior was extended to included stopping (dis-engaging the clutch at very low speed to avoid stalling the engine) and re-starting (re-engaging the clutch

when attempting to accelerate). Several new state variables were added to support the new transition events.

5. A parameter TH_MIN_CL_ACCEL was added to support the automatic re-engagement of a clutch when accelerating from a stopped condition.
6. Upgraded some Import variables to support interactions with internally calculated values via ADD, MULTIPLY, and REPLACE. The Imports are: IMP_AT_CLUTCH, IMP_AV_ENG, IMP_GEAR_TRANS, IMP_IENG, IMP_INV_CAP_TC, IMP_MENG_REACT, IMP_MODE_TRANS, IMP_RM_TC, IMP_R_EFF_TR, and IMP_R_GEAR_TR.
7. Two types of the electrified powertrain model, **Parallel hybrid** (OPT_HEV = 4) and **Full electric** (OPT_HEV = 2), are supported in BikeSim.
8. Target acceleration speed control (OPT_SC = 5) is supported in BikeSim.

Improvements for Moving Object Import Variables

Upgraded some Import variables for moving objects to support interactions with internally calculated values via ADD, MULTIPLY, and REPLACE. The Imports (for object #1) are: IMP_HEAD_OBJ_1, IMP_MSG_OBJ_1, IMP_PITCH_OBJ_1, IMP_ROLL_OBJ_1, IMP_S_OBJ_1, IMP_TYPE_OBJ_1, IMP_VIS_OBJ_1, IMP_V_OBJ_1, IMP_X_OBJ_1, IMP_YAW_OBJ_1, IMP_Y_OBJ_1, and IMP_Z_OBJ_1.

VS Browser: Graphic User Interface (GUI)

64-bit Version of the Browser

The browser BikeSim.exe is a 32-bit application that runs on both 64 and 32-bit versions of Windows. As such, it can load 32-bit plug-in libraries such as the VS Solver bikesim_32.dll but is not able to use 64-bit libraries.

Most users have been working with 64-bit versions of Windows, and many engineering software tools are now available only as 64-bit applications and libraries. For example, the last version of 32-bit MATLAB from MathWorks was 2015b. That means any recent versions of MATLAB and Simulink will work only with the 64-bit VS Solver plug-in libraries.

The 2022.1 release includes two versions of the Browser: BikeSim.exe (still 32-bit) and BikeSim_64.exe (64-bit). The plan from Mechanical Simulation is to drop the 32-bit versions of our tools in the 2023.0 release. (Recent releases have already included both 32-bit and 64-bit versions of the VS Solver libraries, VS Visualizer, and other tools.)

Mechanical Simulation recommends using the 64-bit version unless there is a need to maintain compatibility with 32-bit tools. Given that recent versions of MATLAB/Simulink are only 64-bit, there is slightly better compatibility if the runs made without Simulink use the same VS Solver library as the runs made with Simulink.

Improvements in Existing Library Screens

Existing Library screens were modified.

I/O Channels: Write

The Browser now supports access to pre-processing and post-processing callback functions (I/O Write screen) so the user has the option to execute their own programs before or after the solver is run. As mentioned above, this capability can also be accessed with the VS API.

The filename suffix option has been restored for auxiliary outputs.

Electrified Motorcycles

Two existing library screens are modified, and three new library screens are added to support electrified motorcycles. The library screens are:

1. Powertrain: Shaft Drive (modified),
2. Powertrain: Chain Drive (modified),
3. Powertrain: Hybrid/Electric System (added),
4. Powertrain: Hybrid/Electric Power Management Control (added), and
5. Powertrain: Electric Motor Torque (added).

Miscellaneous Changes

1. Changes were made in Powertrain screens such that unused options are not installed. For example, the `INSTALL_ENGINE` command is not applied for electric powertrains.
2. The **Control: Clutch Shifting Timelines (Closed Loop)** screen was modified to include yellow fields for the new parameters `T_CL_START` and `T_TH_START`.
3. The **Powertrain: Electric Motor Torque** screen includes a new checkbox and a new yellow field for an optional reduction gear. If the checkbox is checked (default is unchecked), the yellow filed appears to set the reduction gear ratio which is used to scale the input and output of the motor torque configurable table, i.e. `SPIN_SCALE_M_MOTOR_MAX` and `MMOTOR_MAX_GAIN`. Also, the gear ratio is used to modify the motor rotor inertia (`I_MOTOR`).
4. The **Control: Shifting (Open Loop)** and **Control: Shifting (Closed Loop)** screens were both changed to allow only three kinds of Configurable Functions that are appropriate for gear as a function of time: Constant, Table (steps, flat-line extrapolation), and Equation.
5. Two more plot links were added to the **Generic VS Commands** screen.
6. The **Tools** menu was modified to clarify the searching of existing runs for uses of the dataset currently in view.

VS Visualizer

VS Visualizer has added a preferences option to force X or Y plot axis labels to show. Users with a small VS Visualizer window and many plots may have hidden axis labels due to automatic plot window scaling. Forcing the axis labels to show will allow users to view VS Visualizer at their preferred window size.

Licensing

The Command-Line License Manager can now run as a Windows Service, allowing for the application to be started automatically when the system is booted. Additionally, running the License Manager as a Service allows for the application to be started, paused, or stopped using the Microsoft Management Console.

Documentation

The following documents were added to the **Help** menu:

1. Powertrain > Electric and Hybrid Electric System (BEV/HEV)
2. Technical Memos > Change Units of VS Math Model Variables
3. Technical Memos > vs_sf VS Connect Server
4. Tools > Database Builder

The following Guide was updated:

5. Quick Start Guide

The following Reference Manuals were updated:

6. System Parameters in VS Math Models
7. VS Browser (GUI and Database)
8. VS Commands
9. VS Commands Summary
10. VS COM Interface
11. VS Math Models
12. VS SDK: The VehicleSim Software Development Kit
13. VS Table Tool
14. VS Visualizer

The following Screen documents were updated:

15. ADAS Sensors and Target Objects
16. Aerodynamics
17. Animator > Bike Shape Assembly
18. Animator > Camera Setup
19. Animator > Rider Shape Assembly
20. Animator > Shapes and Groups
21. Animator > Reference Frames
22. Animator > Sounds

23. Animator > Vehicles and Sensor Targets
24. Brake System
25. Generic Data > Generic Data Screens
26. Generic Data > Generic Table
27. Generic Data > External Parsfile
28. Model Extensions and RT > Custom Forces and Motion Sensors
29. Model Extensions and RT > External Models and RT Systems
30. Model Extensions and RT > Import and Export Variables
31. Model Extensions and RT > Path Detectors
32. Paths, Road Surfaces, and Scenes > Paths and Road Surfaces
33. Paths, Road Surfaces, and Scenes > Road Surface Visualization
34. Paths, Road Surfaces, and Scenes > VS Terrain
35. Payloads
36. Plot Setup
37. Powertrain > Electric and Hybrid Electric Systems (BEV/HEV)
38. Powertrain > Powertrain System
39. Procedures and Events
40. Rider Controls
41. Steering Systems
42. Suspension Systems
43. Tire Models
44. Tools > Atlas GPS Tools
45. Tools > Calculator Screen
46. Tools > Calculator Tool for Tables
47. Tools > VS / ERD File Utility
48. Vehicles > Motorcycles and Rider Bodies
49. Vehicles > Three-Wheel Motorcycles

The following Technical Memos were updated:

50. HPC Licensing
51. Numerical Integration in VS Math Models
52. Validation of VS Vehicle Models

53. VehicleSim License Manager (VSLM)
54. VS Solver Wrapper

The following Real-Time document was updated:

55. RT-Lab Guide

The following SDK documents were updated:

56. The VehicleSim API
57. The VS Vehicle Module Simulation Integration Utility
58. VS Output API: Reading and Accessing VS Output Files
59. VS SDK: The VehicleSim Software Development Kit

Database

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

Electric Powertrain

Two full electric powertrain examples; two pre-transmission parallel hybrid powertrain examples; and three post-transmission parallel hybrid powertrain examples are added under *** Electric Powertrain** category. Those examples involve EPA Urban and EPA Highway battery/fuel economy speed profiles; and open-loop acceleration and re-generative brake maneuver.

Impaired rider

The closed-loop driver model includes a parameter which can be used to delay the application of the calculated steering input by the desired time. New examples have been added which make use of this parameter to model the effects of an impaired driver. The preview time is also shortened proportionally to include an impaired driver's reduced ability to focus on the road ahead.

Parametric sweep example

A new example in the category *** Parametric Sweep** uses the new pre- and post-processing callback feature (I/O Channels: Write, p. 20) to perform a sweep of a parameter of interest. Using a Python script, the run data is duplicated, adjusted, and run with various parameter settings in a pre-processing step. The post-processing step is then used to overlay the results.

Path data from parametric equations using Calculator: Symbolic screen

A new example (**Roads and Intersections** > **Path from Parametric Equations**) shows how the Calculator: Symbolic library screen can be used to convert a parametric plane curve into X-Y coordinates. These coordinates are then used to create a reference path for the bike to follow.

Speed controller

A target acceleration speed control example which utilizes speed controller option of `OPT_SC` = 5 is added under *** Speed Controller** category.

VS Command Examples

Copies of the Quick Start run were made to add outputs generated using new VS Commands Inverse (inverse of a Configurable Function) and Partial (partial derivative of a Configurable Function expression).

BikeSim 2022.0 New Features

VS Math Models

GPS Calculations

GPS coordinates are calculated and provided as output variables for the first vehicle unit and for moving objects. The conversion from global X and Y coordinates in the simulation model coordinate system to GPS have been based on the starting location of the vehicle, with updates occurring when there is a significant change in GPS latitude. Several improvements were made to accommodate simulations involving multiple vehicles and moving objects that might be separated by significant distances.

1. The GPS conversion parameters `GPS_REF_LAT`, `GPS_REF_LONG`, `GPS_REF_X`, and `GPS_REF_Y` now retain their initial value, reflecting the value associated with the creation of the road or scene. Instead, when any reference point is reset for a vehicle or moving object, the run's log file will contain a line indicating the vehicle or moving object ID and the latitude/longitude at the reset.
2. The GPS reference point is now checked at initialization, to improve GPS output accuracy in cases where the initial location of the vehicle or moving object is past `GPS_RANGE_Y`.
3. Moving object GPS output calculations now account for the reference point reset implied by the parameter `GPS_RANGE_Y`, rather than always using the reference point established by `GPS_REF_LAT`, `GPS_REF_LONG`, `GPS_REF_X`, and `GPS_REF_Y`.
4. Moving object GPS output calculations now function if the X-Y coordinates of the moving object are specified directly. Previously, moving object GPS outputs were only produced if the moving object was set to follow a path.

Miscellaneous

A new system parameter `OPT_ECHO_DEFAULT` is available to disable the identification with the indicator [D] in the Echo file for default values that were not set by reading an input file. This can be helpful in advanced applications, such as when a new run is made using an Echo file written at the end of a previous run. In this case, the [D] indicator never appears in the Echo files for the continuation run. The new parameter may be used to ensure the [D] indicator doesn't appear in other runs either, simplifying the use of text editors to compare files.

VS Browser: Graphic User Interface (GUI)

Tool: Find links to the current dataset from any library

A new command was added to the **Tools** menu for searching: **Find All References to this Dataset in "Run_all.par" Files by Library...**

The existing **Find All References to This Dataset** option finds datasets that are immediately referenced by others. The search can be repeated, to find datasets that reference datasets that reverence the current dataset, but this form of manual repeating is time consuming.

The new search option takes advantage of `Run_all.par` files that are made automatically whenever a run is made, or VS Visualizer is used to view results. This has significant advantages:

1. Searching these files is rapid compared to searching the entire database. Even with large databases, the search is much less than a minute.
2. The searching finds all references where the current dataset was used in an existing run.
3. Results are filtered to show only cases where the current dataset referenced (no matter how indirectly) by datasets in a specified library.

For example, perform a search from a **Tire** dataset to find all the **Run Control** datasets that may somehow make use of the current tire dataset. Or, search for all vehicle datasets that used the current tire dataset in a run.

Miscellaneous Changes

- Users can now specify the results output directory of an FMU with self-contained mode enabled.
- Many minor changes were made to fix bugs, correct typos, improved consistency, etc.

VS Visualizer

Added new option to scale plots non-symmetrically by pressing the "alt" key while dragging with the middle-mouse-button (or left+right mouse buttons). In this mode, moving the mouse up/down zooms vertically, moving left/right zooms horizontally. There is also an option on the plot window right-click context menu ,"Asymmetric mouse zoom", to switch which mode is default (active w/o the Alt key).

VS Solver Wrapper (Command Line Interface)

Support of VS Solver Wrapper with a command-line interface has been improved to provide more options when using external software to provide automation.

- VS Solver Wrapper can now execute simulations using Simulink completely using the command line interface. Simulink will be automatically launched when the `SIMULINK_MODEL_FILE` keyword is detected in the "`all.par`" input file. For more information, see VS_SolverWrapper.pdf.
- VS Solver Wrapper can execute simulations of an FMU using the command line interface. An FMU must be FMI version 2.0 and self-contained that includes all input

files and binary solvers: vehicle solver 32/64 bit for Windows and Linux 64 if running on Linux 64, `simfile.sim`, `Run_all.par`, events parsfiles, vs_terrain file, external tire files, see VS_SolverWrapper.pdf.

- VS Solver Wrapper can now load and run External Tire model datasets. Previously these were referenced through absolute path, due to third-party requirements. They are now referenced through relative path, with the absolute path being determined dynamically through the PROGDIR simfile parameter.

Documentation

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

1. High Performance Computing (HPC) and VehicleSim

The following Reference Manuals were updated:

2. VS Browser (GUI and Database)
3. VS COM Interface
4. VS Commands Reference Manual
5. VS Commands Summary
6. VS Math Models Reference Manual
7. VS Visualizer Reference Manual

The following Screen documents were updated:

8. ADAS Sensors and Target Objects
9. Animator: Wheel Arrows and Other Indicators
10. External Models and RT Systems

The following Technical Memos were updated:

11. High Performance Computing (HPC) and VehicleSim
12. HPC Licensing
13. VS Camera Sensor Simulink Block
14. VS Solver Wrapper

The following Real-Time and DS System document was updated:

15. dSPACE RT Guide

New Animator Resources

New Animator Shape Assemblies have been added for the three-Axle Sleeper Cab, Flatbed Trailer, Shipping Container Hauler, and 53-ft. Box Trailer.