

The New Features of BikeSim 2

Development began in 2005 to adjust the popular simulation environment shared by the Mechanical Simulation software products CarSim and TruckSim to take full advantage of the many technological improvements in computer hardware and software that have taken place since the original development in the early 1990's. Major improvements were made in vehicle simulation technology, and these improvements are now included in BikeSim with the release of version 2.

As with any major software update, there are hundreds of new features, bug fixes, and minor improvements in various aspects of the software parts. However, the focus of this release is the support of a new architecture, which in turn will enable many other improvements over the next few years.

This memo describes the new architecture used for BikeSim 2. It is technical and is intended for readers who are already familiar with the features of BikeSim 1.

Background and History

Vehicle simulation using numerical methods has developed considerable since research in the 1970's when early motorcycle models were created and validated.

Vehicle Simulation and Multibody Dynamics

Prior to the 1980's, most vehicle math models with a complexity approaching the models in BikeSim, CarSim, and TruckSim were written by hand for use on mainframe computers. In the late 1970's, multibody computer programs such as ADAMS® were developed and applied to the simulation of vehicles, at the expense of requiring much longer computer times. In the late 1980's symbolic multibody programs were developed that could generate computer source code for multibody systems such as ground vehicles. Perhaps the most advanced was a program called AutoSim, developed by the author (Sayers) at University of Michigan Transportation Research Institute (UMTRI). AutoSim was validated and used to replace most of the car and truck simulation programs used at UMTRI at the time. AutoSim was also licensed to the American car manufacturers and some universities and research labs around the world. The custom simulation programs generated with AutoSim have the high efficiencies associated with hand-written custom programs, with far better reliability.

Since the founding of Mechanical Simulation Corporation by Sayers, Dr. Gillespie, and others in 1996, AutoSim has been maintained, updated, and used to create all of the math model solver programs in CarSim, TruckSim, and more recently, BikeSim.

The Simulation GUI

In the early 1990's, a GUI and database design for simulation were developed by Sayers and others at UMTRI to provide a more productive software environment for all users, from novices to specialists. The database and GUI design have been maintained and improved over the past twelve years at Mechanical Simulation, along with tools for automated plotting and 3D animation.

Co-Simulation

In the late 1990's many users combined software using "co-simulation" to build models from several packages. A popular combination was CarSim or TruckSim and MATLAB®/Simulink® (from The Math Works). Other combinations involve real-time testing systems from companies such as dSPACE, Opal-RT, and others. The old idea that a computer is used to run one vehicle math model with all features for all users is obsolete; it is now essential for many programs to run simultaneously and communicate easily with each other, to allow users to configure the simulations for many unique requirements.

BikeSim

Many of the modeling concepts used in the BikeSim math models were developed and validated by Professor Robin S. Sharp through research that began in the 1970's. The equations for those early models were derived by hand for linearized analysis and nonlinear simulation. In the 1990's Sharp switched to AutoSim to provide models in support of research at Cranfield University and later Imperial College London in United Kingdom in collaboration with Professor Limebeer and others. In 2002 Imperial College provided the AutoSim-based model to Mechanical Simulation to support the transfer of technology from academia through the development of a commercial software product. David Hall at Mechanical Simulation developed the first beta version of BikeSim (available in 2003); Dr. Yukio Watanabe continued the development through the release of BikeSim version 1.0 in 2005.

VehicleSim Capabilities

In 2006 the original AutoSim code generator was replaced with a new system called VehicleSim® (VS) Lisp. At the same time, the database browser design was revised to work better in the Windows environment. The database design is now called a VS database, and the main browser program (`bikesim.exe`) is called a VS browser.

CarSim 7, released in 2007, was the first commercial product based on VehicleSim technology. The same VS technology supports BikeSim 2. Some key features of VehicleSim are:

1. The math models maintain the same high efficiency associated with models from past versions, allowing real-time operation for hardware in the loop (HIL) testing and driving simulators.
2. The models are extendible by users. Some of the symbolic capabilities previously associated with AutoSim are now available at run time, allowing users to add new output variables, controllers, and automation options.
3. The models communicate with a standard application program interface (API) that is well documented and can be used to connect with nearly any simulation environment.
4. The main platform for VehicleSim programs is Windows OS. Although the solvers are also provided for other operating systems such as Linux and QNX, Windows is the standard platform for engineers, and VehicleSim programs operate as expected by the corporate IT departments for Windows machines and networks.

BikeSim 2 Achievements

The highest priorities in preparing BikeSim 2 were to complete the new architecture for the solver programs, and improve the browser and other parts of the software as needed to support the new solvers. The new capabilities achieve these objectives for BikeSim 2:

1. The software is **backward compatible** for existing BikeSim 1.0x users; existing datasets work in BikeSim 2.
2. For new users, the user interface and documentation make BikeSim **easier to learn**.
3. For advanced users, VehicleSim technology makes BikeSim **easier to extend and automate**.

VehicleSim Solver Program Architecture

The original AutoSim symbolic code generator derived equations of motion for a multibody system in symbolic form and wrote them in the Fortran computer language. The equations were ordinary differential equations (ODEs) that are solved with numerical integration methods. Later, AutoSim was extended to generate code in other languages, such as MATLAB, ADSIM, ACSL, Maple, and C.

Since about the year 2000, all solver programming at Mechanical Simulation Corporation has been done in the ANSI C language, which is supported by all of the RT systems used for BikeSim. VS Lisp was developed to generate source code specifically for ANSI C. Working in one language allows an architecture for the VS solver program that is much more ambitious than the original AutoSim-based solver design.

Nonlinear Tables

Many of the vehicle properties, controls, and road properties are described with nonlinear tables. In past versions, the form of table interpolation was determined by the developers and hard-coded into the VS solver. The new architecture supports run-time selection of one of many possible calculation methods, depending on user needs. These range from a constant, to a linear coefficient, to linear interpolation, to a spline, to 2D carpet plots. (There are currently 11 calculation options.)

Most tables include a scale factor and offset that can be used to transform a given nonlinear shape. These have keywords to help integrate with other software (DOE, sensitivity, optimization, etc.).

Communicating with Model Extensions

The desire to extend models with Simulink, other simulation environments, or custom routines written in C has led to more attention to communication and integration of the “native” (compiled) equations with equations provided by users.

Communication with other software is accomplished with arrays of import and export variables. The VS solver equations can include effects of variables that are defined in other software and imported as the run proceeds. Export variables are computed by the VS model and can be sent to other software during a run.

With past versions, the internal equations were usually set up to add the import variables to internal values. For example, an imported road friction would be added to the friction calculated internally using tables. In the new architecture, most import variables are closely tied to internal native variables. These imports can function in any of four modes

1. ignore the imported variable,
2. replace the native variable with the imported value,

3. add the imported variable to the native variable, or,
4. multiply the imported variable by the native variable.

For example, road friction specified in detail using existing BikeSim tables can now be scaled during a run using a scale factor imported from Simulink (option 4). As another example, it can be replaced by a friction coefficient completely defined in Simulink (option 2).

VehicleSim Commands and Processing

The new VS solvers support a symbolic calculator that allows inputs to be expressed either numerically (as in past versions), or in terms of equations involving other model parameters and variables.

The solver programs also include a preprocessor with advanced commands that allow users to customize and extend the models by adding equations at run time. Import variables can be assigned to equations involving other BikeSim variables. For example, the imported road friction variables for each tire can be defined with equations defined at run-time that are based on other variables in the model.

New variables can be defined at run-time for a variety of purposes: import, export and output, constant parameters, or auxiliary variables used in other equations.

New variables added with VS commands can be calculated with equations that are also defined with VS commands. The equations can be algebraic, involving all existing and new variables and parameters in the model. Further, algebraic equations can be defined at run time for the derivatives of the new variables, allowing users to add degrees of freedom via differential equations.

The event capabilities from past versions of BikeSim have been extended to include algebraic equations for defining thresholds, and for defining new parameters and controls. For example, an event can now be triggered when a complicated algebraic equation involving several output variables is satisfied. When the event occurs, controls can be defined using scaling based on algebraic combinations of current variable states. For example, the scale factor for a table for steering can be defined by an equation involving a level of lateral acceleration and speed.

Functions that can be used in VS command equations provides access to the 3D geometry of the BikeSim roads, making it easy to define extra traffic vehicles that follow the road, or advanced rider models that look ahead at the road geometry.

The units associated with parameters and variables can be set at run time. For example, set dimensions in inches instead of millimeters. Changes can be made for individual variables or the entire model. New units can even be defined at runtime (for example, you might choose to plot velocity in furlongs per fortnight).

The capability of extending BikeSim models with VS commands is part of every BikeSim package, including all RT versions.

The VehicleSim API

The VS solvers are Windows DLL modules with the VehicleSim application program interface (VS API). The same solver DLL is used in all Windows simulation environments (built-in, Simulink, LabView, ASCET, command-line EXE, etc.). When no other software is used, the DLL's are loaded and run directly by the BikeSim browser. Support for other software such as

Simulink and LabView is handled by small “wrapper programs” that load and run the VS solver and communicate as needed with the other software.

All of the capabilities of the VS commands for extending the built-in model can also be implemented using the VS API, such as adding variables, adding differential equations, etc.

Example source code (ANSI C) is provided showing how to use the VS API to run the VS solvers from programs written in any language that can load a DLL (C, C++, Visual Basic, Excel (with VB macros), etc.). Examples are also provided showing how to extend existing models with custom C code, with the equivalent of the new VS commands.

Numerical Integration

The VS solvers now support five methods for numerically integrating ordinary differential equations (ODEs), with all methods being suitable for RT applications.

New Multibody Model Features

More detail has been added to the vehicle modeling to extend the range of valid simulation conditions. Some of the changes are:

- TNO Delft-Tyre model and nonlinear table lookup tire models were added.
- The interaction between the steering system and the suspension were redone with more detail for accurate 3D kinematics over the full range of steering.
- The steering system has two options for caster angle geometry: fixed relative to the main frame, or variable with suspension stroke.
- The suspension models have full nonlinear kinematical behavior that can support various types of suspension kinematics. For example, the front suspension model can handle *telescopic fork*, *McPherson strut*, *double wishbone*, *springer*, *bottom link*, etc; and the rear suspension swing arm can work with or without *parallel link*.
- The front suspension now has compliance in longitudinal (bend) and roll (twist) relative to the main frame.
- The powertrain components have been organized to more easily replace parts of the powertrain with external models.
- A dynamical time lag (degree of freedom) was added to the engine model.
- Driveshaft torsional stiffness and damping were added.

VehicleSim Browser Graphical User Interface

The BikeSim GUI is even easier for new users to learn, while supporting many more options for advanced users.

Sidebar for Navigation and Notes

The user interface window now has a sidebar that shows an active tree-based view of the database, with instant access to any of the 100+ datasets that might be used in the current run.

The sidebar also shows notes about every dataset, providing a visible and convenient place for user-defined documentation.

Automation and the COM Interface

The server capability of past versions has been extended and re-written to offer full support via the Windows COM interface. Users can control the SGUI from other programs such as Excel, Visual Basic, MATLAB, C/C++, Python, etc. to provide automation for design of experiment, sensitivity, and other applications.

Installation and Windows Support

The software follows the organization standard for Windows (with programs and data in different areas), for better management by corporate IT departments. The new organization makes it easier to work with multiple BikeSim databases from a single installation on the same computer.

Users can still specify custom configurations such as network database locations.

Specifying Tables

Many vehicle properties, controls, and road properties are described with nonlinear tables whose properties can be specified in the datasets. The GUI has been extended to support new features of the VS solvers, and to improve the visualization and editing options in the VS browser.

- Nonlinear tabular data can be shown and edited in both spreadsheet views and classic plain text.
- Graphs of tabular data show the interpolation and extrapolation of the data. When spline interpolation is used, the original data points are shown along with the smoothed interpolation.
- Most nonlinear tables include a drop-down list to specify the type of relationship, ranging from constants, to linear coefficients, to linear interpolation, to spline, to 2D carpet plots. (There are 11 calculation options.)
- Most table screens include a symbolic calculator tool, to rapidly generate data from algebraic equations, or transform existing series of numbers.
- A new Symbolic Calculator screen provides a library for using and storing table equations.
- Most tables include a button to transfer the contents to Microsoft Excel for editing and transformations.

New and Improved Screens

A number of existing screens have been improved to support new features of the solvers and to simplify integration with external software.

- The layout of the main Run Control screen has been simplified for new users, while offering more options for advanced users.
- A new Events and Procedures screen organizes data for standard tests, and provides a clean interface for defining events.
- The Model setup screens include more simulation settings for advanced users.

- The import and output control screens have been extended to support new options.
- The powertrain screens have been revised to simplify the replacement of components with external models.
- The tire screens have been revised to support multiple tire model options.
- All new features in the VS solvers are supported by corresponding new data screens (steering torque input, steering/suspension interactions, etc.).

Animator

The animator has been re-written to support multiple live (real-time) animations on multiple monitors with high-frequency full-screen displays. Although the work was done to support live animation for driving simulators, the improvements are also available for regular post-processing animation. Other new features have also been added.

- The animator can generate AVI and other standard Windows multimedia files for use in PowerPoint and other software.
- Audio options have been added for engine noise, wind noise, and tire noise in driving simulators.
- Overall efficiency has been improved to allow high-frequency update rates with much less CPU demand.
- Dynamic transparency is now supported, and the number of “ghosts” has been increased to support the display of tire skid marks and other visualized traces of vehicle behavior.
- Communication for live animation has been improved to provide high-frequency full-screen displays on multiple monitors as needed for driving simulators.
- The animator automatically sets the best refresh rate for the computer hardware.
- Improvements have been made for fog, certain lighting conditions, and field of view.
- Example vehicle shape files have been improved to provide exceptionally high-quality visual presentations.

Documentation

As the BikeSim capabilities have grown and matured, the documentation organization has been adjusted to provide more information and to make the information easier to access. Over 50 reference documents are provided with BikeSim, to provide all levels of details over a range of topics.

- The main reference manual has been converted to a modular form (about 40 specialized documents on topics such as “Suspension Systems,” “Tires,” “3D Roads,” etc. This reduces the amount of redundant material and allows greater detail to be added.
- Reference documents that list available import and output variables are available in text (as before) and in Excel, where they can be sorted by name or category.

- A new reference manual describes the VehicleSim solver program design, including the VS calculator and command language.
- A programmers reference manual is provided that describes the VehicleSim API.
- Numerous technical memos covering model details and examples are provided in the Help menu.

Future Plans

The new VehicleSim architecture provides tremendous capabilities for extending and customizing BikeSim. Many of the customer requests for model improvements made over the past few years can now be accommodated. However, it will take time to appreciate and make use of some of the new features.

Expect to see more upgrades in the coming months, as we extend the solver programs to make use of the new capabilities. Also, look for more downloadable examples in the Users section of our web site that demonstrate and explain more applications of the new capabilities.

Mike Sayers, Ph. D., CEO and Chief Technology Officer

October 6, 2008