bikeSIM®: Math Models

BikeSim math models represent the dynamic behavior of two- and three-wheeled motorcycles. The VehicleSim® (VS) Math Model architecture is used.

BikeSim VS Math Models are built using dynamically linked VS Solver library files, available for 13 operating systems: Windows (32- and 64-bit), Linux, and real-time platforms used for hardware-in-the loop. The models work well with other software (Simulink, LabVIEW, FMI, ETAS ASCET, EPIC Unreal, Custom programs, etc.) for automation or extensions to the models.

A basic BikeSim model runs more than 10 times faster than real time on a typical Windows computer.

Multiple vehicles may be simulated simultaneously and communicate with each other using external software.

Vehicle Math Models

Configurable Table Functions

- Potentially nonlinear relationships between variables are defined with VS Configurable Functions that can be:
    - Constants
    - Linear coefficients
    - Nonlinear tables with several interpolation methods involving one or two independent variables
    - User-defined formulas
- Configurable Functions include offset and gain transform parameters for dependent and independent variables.
- There is no built-in limit to the length of tables.

VS Reference Paths

- A VS path defines an S-L coordinate system (S = distance along path, L = lateral distance from path).
- Each path is a sequence of segments, where each segment may be: straight, an arc, a clothoid, or an X-Y table.
- VS paths are used for rider controls, locating traffic vehicles, and defining 3D road properties.
- A VS Math Model supports up to 500 VS Paths.

Rider Controls

- All rider controls can be handled by built-in controllers, defined by equations added with VS Commands, or imported from external software.
- The built-in rider model can steer to follow a target path, which can be changed during the run.
- A Closed-Loop Lean Target (for the bike) can be combined with a Rider Body lean angle.
- Steering can also be accomplished using open-loop rider body control and open-loop handlebar torque.
- The rider model can control speed based on target speed and acceleration limits, curvature of the target path, and 3D road geometry (banking, grade, curvature).
- Gear shifting and clutch controls can be handled with shift schedules and automatic throttle-clutch interactions.
- Closed-loop and open-loop controls can be combined to simulate intervention systems.
- Open-loop braking is represented as lever force (front and rear).

3D Road Geometry and Friction

- The 3D ground surface includes 3D geometry, friction, and a tire rolling resistance coefficient.
- The 3D surface may be a set of VS Roads or VS Terrain.
- Up to 200 VS Roads may be built with components:
    - VS Reference Path for S-L coordinate system.
    - Configurable Functions for elevation and friction using S-L coordinates and variable-width tables.
    - Boundaries to connect adjacent VS Roads.
- VS Terrain provides a single mesh-type ground surface, created with VS Scene Builder with several options:
    - Create interactively by dragging 3D Tiles.
    - Import datasets from OpenDRIVE format.
    - Import 3D FBX files from other software.
- Road profiles “wander” to follow the vehicle tires, providing high-frequency road roughness inputs. Road profiles are measured routinely by some road agencies.

Wind and Aerodynamic Effects

- Six aerodynamic forces and moments are applied to the sprung mass.
- These forces and moments are shaped by Configurable Functions of aerodynamic slip angle.
- Ambient wind speed and heading can be set with tables, runtime equations, or imported from other software.

Suspensions

- The suspension models have full nonlinear kinematical behavior.
- Front suspensions can be with or without compliance in the longitudinal direction (bend) and roll (twist) relative to the main frame.
- The front suspension can be many types: *telescopic fork*, *McPherson strut*, *double wishbone*, *springer*, *bottom link*,

etc. The detailed kinematic motion can represent *anti-dive* geometry.

- Rear suspensions can be a swing arm with or without a parallel link. The model specifies detailed kinematic motion of the wheel hub to represent *anti-jacking* geometry.
- The rear suspension can be with or without compliance in the lateral direction (bend and steer) and roll (twist) relative to the main frame.
- Each wheel moves vertically. Longitudinal movement and dive angle of wheel hub are related to vertical position by nonlinear tables.
- Suspension springs are nonlinear and include hysteresis due to friction.
- Damper forces are nonlinear functions of stroke rate.
- Both the spring and damper use nonlinear lever ratios.
- Front *lean-linkage* suspension for the three-wheeled motorcycles is included.

Steering System

- The interactions between the suspension, steering, tire, and ground are handled with a detailed multibody model with steering axis.
- The steering system geometry is parameterized by caster angle, wheel axle height, fork length, and fork offset.
- The steering system includes mechanical limits and damping.
- Caster angle can be fixed relative to the main frame or variable with suspension stroke.

Brake System

- Master cylinder pressure is calculated by the lever/pedal force input through the booster mechanism.
- Brake torque is calculated by actuator pressure and disc data representing the area, effective radius, and pad coefficient of friction.
- The brake system can involve external programs (e.g., Simulink) to provide advance control such as ABS, TCS, and stability control.
- Special equations handle wheel lockup to obtain the correct reaction torque and avoid numerical instability.

Tires

- BikeSim includes several tire models, along with a program interface that supports external tire models, such as MF-Tyre / MF-Swift from Siemens/TNO.
- BikeSim runs with MF-Swift from Siemens/TNO and FTire from COSIN (extra licenses are required from Siemens and COSIN, respectively, to use their models).
- External tire models can apply forces at either the ground contact point or the wheel center.
- Different tire models can be applied to the front and rear wheels.
- The original tire model uses the *Magic Formula* to represent longitudinal force, lateral force, and aligning moment as functions of slip, load, and camber. The shear forces are applied at a single contact point which moves around the tire circumference and laterally around the side wall: this automatically defines overturning moment.
- An internal table look-up model uses fully nonlinear asymmetric tables to represent lateral force, longitudinal force, and aligning moment as functions of slip, load, and camber.
- As part of the internal table look-up model, overturning moment due to the tire contact kinematical effect can be replaced with non-linear tables as functions of slip, load, and camber.
- Variable friction conditions are handled using similarity, allowing BikeSim to maintain both linear and limit properties of the tire (for table look-up model).
- Transient effects of rolling are included using relaxation length. Relaxation lengths can be constant or defined as nonlinear functions of vertical force and slip.
- Special equations are used to maintain realistic tire behavior at low speeds when the assumptions of a rolling tire are not valid.

Powertrain

- BikeSim has detailed powertrain models for chain drive and shaft drive. There is also a minimal model used for speed control in which torque is applied directly to the wheel(s).
- Engine torque is defined with a 2D table that relates torque to throttle input and crankshaft angular velocity (RPM).
- The engine feeds torque to the transmission through either a mechanical clutch or a hydraulic torque converter with a primary gear.
- The transmission converts torque and speed based on the current gear selection, with spin inertias and efficiencies that depend on the gear selection.
- Continuously variable transmissions (CVT) are supported.
- The torque from the transmission goes to either a sprocket and chain mechanism or a driveshaft.
- The chain has tensional stiffness and damping. The force vectors for driving and engine braking affect the swing arm motion.
- The driveshaft has torsional stiffness and damping.
- Fuel consumption is defined with a 2D table.

Sensors and Traffic

- The models include several kinds of virtual sensors that detect various types of vehicle motion, including acceleration, speed, and jerk.
- Up to 200 moving objects can be added that are updated automatically to convert simple path-based commands into full 3D geometry.
- Motion of an object can be constant, set by specifying speed, controlled by acceleration (simple physics), set with algebraic equations, or imported via import variables.
- The objects can be recycled for extensive runs, to reappear after they go out of view.
- Objects that move based on speed or acceleration support off-tracking, for realistic low-speed traffic turns.
- Objects used to represent traffic vehicles support brake lights and reverse lights.
- Objects may be rectangular, circular, segment (e.g., signs), or polygonal.
- Segment objects have a limited viewing angle, to mimic signs and signals with limited visibility.
- Up to 99 ADAS range and detection sensors can be included that detect the moving objects. An optional license is needed for sensors (but not for objects).
- Each detection includes 24 variables that can be exported to external controllers (e.g., ADAS).
- Objects can block each other (occlusion). The sensor detection variables respond only to the portion of the object that is within the field of view.
- Detection sensors can be placed on the vehicle or moving objects (to detect and simulate collisions).
- Objects may be attached to vehicle sprung masses to support ADAS simulations with multiple vehicles or provide details of collisions with pedestrians.

VS Math Model Input and Outputs

Input Data Files

- BikeSim reads all input from text files that are normally generated automatically by the Browser/GUI. These files can also be made externally for advanced applications.
- Input files for BikeSim follow a simple keyword-based format called the Parsfile. BikeSim can recognize thousands of keywords when processing input files.
- Each input line can optionally specify alternate units for a parameter.
- Values can be assigned directly to model parameters with numbers, numerical expressions (e.g., 1/16), or symbolic algebraic expressions involving other model variables.
- Parsfiles support the INCLUDE capability, allowing advanced applications such as design of experiments (DOE), sensitivity, and customized automation methods.

Output Variables

- BikeSim generates from about 400 to thousands of built-in output variables, depending on whether there are sensors, traffic vehicles, etc.
- A subset of the available outputs can be specified at run-time, to control the size and organization of output files.
- Writing to file can be enabled and disabled during the run, to save only interesting results from long simulations.
- BikeSim provides a GUI for browsing the lists of available variables, sorting by several categories.
- All variables are described in documentation files in both text and spreadsheet format.
- Output files may be written in several binary forms (32-bit and 64-bit) or CSV (text) spreadsheet format.
- Output variables are used for several purposes:
  - Make plots that show vehicle behavior.
  - Motion information for video visualization.
  - Input to other post-processing software.
  - Export to other software during the simulation.

VS Commands and Python

VS Solvers include the VS Command scripting language for customizing the model and its operation. VS Commands are supported in all versions, including real-time systems.

- VS Commands can add new equations at several locations in the sequence of simulation calculations.
- VS Commands can add new parameters, output variables, and state variables as needed to extend the model.
- VS Commands can add new differential equations.
- VS Commands can add new functions to simplify other formulas or series of equations.
- VS Commands can define new units.
- VS Events monitor custom formulas to trigger the reading of a new Parsfile to change values, modes, etc. This is used to script complicated procedures.
- The Windows and Linux versions provide embedded Python in support of full programming options.

Working with Simulink® and External Software

- The BikeSim VS Math Model is made with functions from a dynamic library file.
  - The BikeSim GUI runs VS Math Models directly.
  - BikeSim includes MATLAB/Simulink S-Functions.
  - BikeSim works with LabVIEW.

    - BikeSim can generate functional mockup units (FMU) to run under the functional mockup interface (FMI).
- MATLAB, Visual Basic (VB), and other languages can operate the Browser/GUI using Windows COM.
- BikeSim has a `LINEARIZE` command to generate linearized A, B, C, and D matrices for use in MATLAB.
- The VS SDK (software development kit) is available for Windows and Linux. It includes numerous application program interfaces (APIs):
    - The VS Solver API is used to run and interact with the VS Solver from C/C++ and languages that can load a DLL file (Python, MATLAB, VB, etc.).
    - The STI API helps connect external tire models.
    - The Shared Camera Buffer API accesses 3D information from VS Visualizer cameras.
    - VS Output API helps read and write output VS files.
    - VS Table API helps interact with Configurable Functions.
    - VS Terrain API works with VS Terrain files.

Import Variables

- Calculations from external models and measurements from hardware-in-the-loop (HIL) can be imported into BikeSim. These include most forces and moments, fluid pressures, controls, ground geometry under each tire, etc.
- The vehicle models can import values for hundreds of built-in variables.
- Most of the import variables can be combined with native internal variables with one of three modes:
    1. replace the native variable,
    2. add to the native variable, or
    3. multiply with the native variable.
- BikeSim provides a browser for activating import variables from the lists of all those that are available.
- New import variables can be defined with VS Commands to pass through data from other software. E.g., variables from Simulink can be passed through to the animator.

Export Variables

- All variables available for writing to output files are also available for export to Simulink or other external code.
- Variables are exported only if activated at runtime, as needed to be compatible with the external model.

Multibody Model Specifications

State Variables and Degrees of Freedom

BikeSim has ordinary differential equations (ODEs) for the dynamics of multibody systems, including rigid bodies, fluids, tires, controllers, and other dynamic parts. Additional state variables are used to define the state of the model for features such as friction, clutch slipping, controllers, etc.

- The number of ODEs and state variables depends on many options available in the model. The VS Solver is used to generate a list of all state variables for any given simulation setup.
- The basic fixed-caster BikeSim model has 53 ODEs and a total of 94 state variables.

Equation Form

- The equations of motion are derived from first principles for 3D motions of multiple connected rigid bodies, using Kane's equations for the multibody dynamics and constraints.
- The equations of motion are ODEs that are not stiff.
- The built-in VS library provides six methods for solving the ODE's (Adams-Bashforth, Adams-Moulton, Runge-Kutta, and Euler methods).
- All methods run at a fixed time step and may be used for real-time HIL applications.
- The algorithms work well with measured and sampled data sources, even when there are discontinuities.
- The Solver libraries are compiled with extensive optimizations for efficient use either alone or with other software (e.g., Simulink, LabVIEW).

Initialization and Restarts

- BikeSim supports many initialization options, from automatic to detailed specification of any state variable.
- The complete state of the vehicle model is saved at the end of each run, to support continuation of advanced automation and optimization methods.
- The state of the model can be saved during a run and fully restored during the run, in support of advanced optimization methods and repetitive test sequences.