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

Validation of VS Vehicle Models

Michael Sayers, Ph.D., Chief Technology Officer

Vehicle models are typically validated by comparing outputs calculated by the model to similar outputs from a reference. This memo gives some background on concepts of usefulness and validation for vehicle dynamics models.

The most common validation that is done for a new version of a tool that has existed for years is to compare results from the new version to results from a previous version in which similar behavior is expected. This memo shows methods used routinely at Mechanical Simulation for comparing outputs from different versions of the math models using VS Visualizer. It shows the levels of agreement that can be expected between different versions of CarSim, TruckSim, and BikeSim, and discusses several sources of model disagreement.

Some examples included in CarSim and TruckSim that can be useful for evaluating new vehicle datasets are described. These tests can be helpful in confirming that a simulated vehicle is meeting expectations with respect to a physical vehicle of interest.

Usefulness of a Vehicle Dynamics Simulation Tool

The statistician George E. P. Box famously wrote that "essentially, all models are wrong, but some are useful." For the physics-based models in CarSim, TruckSim, and BikeSim, the term "useful" means that the software helps engineers and other technical people make informed decisions based on vehicle behavior in conditions and scenarios of interest.

The usefulness of a software tool that simulates vehicle dynamics depends on two broad factors:

1. Can the software tool predict physical behavior as measured in a specific test of interest, within an acceptable tolerance?
2. How much effort is needed to use the software tool to simulate tests of interest?

The first factor is often stated simply as: "is the simulation valid?" The second might be restated as "how much effort does it take?"

When taking both factors into account, there is a wide range of trade-off between accuracy and convenience. Many levels of detail and complexity are employed in making decisions involving vehicle dynamics behavior, ranging from simple to highly complicated. Here are some popular types of tools:

1. Spreadsheets with simple calculations based on quasi-static conditions.
2. MATLAB or Simulink models with a few degrees of freedom (DOF) used to test controller concepts.
3. 1-D custom math models used to simulate straight-line powertrain tests.
4. 3D system-level vehicle models such as CarSim, TruckSim, and BikeSim, capable of reproducing vehicle motions for a full range of handling, braking, and powertrain tests based on driver controls.
5. 3D vehicle models with more detail in some areas of interest (detailed fluid flow for aerodynamics, component details such as parts of a steering system, flexible bodies, etc.).

Because many tools in the above categories are currently used in automotive development, all qualify as "useful" for some applications. This also implies that all approaches can be "valid" so long as it is understood that the concept of "valid" applies to the specific application of the tool.

CarSim, TruckSim, and BikeSim use multibody physics models to simulate the dynamic behavior of highway vehicles in response to controls from the driver or rider, on-board controllers, 3D ground geometry, and aerodynamics. These tools are typically applied in the following areas:

1. Vehicle design and development. OEMs apply simulation to reproduce established vehicle dynamics tests for evaluating designs, optimizing components, and evaluating controllers and components provided by suppliers. Suppliers apply simulation to the development and

testing of on-board controllers, and evaluation of alternate designs for a customer OEM vehicle.

2. Regulatory compliance. OEMs perform thousands of simulations to ensure each vehicle system will comply with safety regulations involving braking and stability. Regulations such as ECR-R13H encourage the use of simulation as an alternative to physical testing.
3. ADAS and autonomous driving scenarios. Simulations involving an ego vehicle encountering other "actors" such as traffic vehicles, pedestrians, and others, interacting with complicated road intersections, traffic signals, and other factors that complicate the scenarios. OEMs might perform millions of simulations covering many combinations of interactions.
4. Driving simulators. Scenarios that require evaluation by human driver are performed with driving simulators that are realistic enough to evaluate differences in candidate designs. Simulators may also be used to evaluate passenger perceptions of autonomous driving methods.

Validation definitions are always tied to specific applications. In each of the above areas, a tool is "valid" if it is deemed useful. The criteria for the first two categories are tied to specific tests, so good agreement can typically be obtained between the simulation and the testing if the inputs to the model have been obtained with the required accuracy. The third and fourth categories typically cover a wide range of conditions that have not been tested. The expectation is typically that the simulation behavior is close enough to physical behavior that the behavior seen in the many scenarios is representative of what would be found in the physical world in similar scenarios.

Validation by Comparison with Physical Testing

When simulation is used to predict the behavior that would be measured in specific physical tests of the vehicle of interest, a full validation for a tool includes both factors mentioned earlier: getting information about the vehicle and procedure of interest as inputs to the tool and exercising the tool to predict behavior that will also be measured in physical testing.

ISO Simulation Validation Standards

The International Organization for Standardization (ISO) has recently been publishing formal validation methods for vehicle simulation tools. The first two that apply for VS Math Models are ISO 19364 "*Passenger cars — Vehicle dynamic simulation and validation — Steady-state circular driving behavior*" (2016) and ISO 19365 "*Passenger cars — Validation of vehicle dynamics simulation — Sine with dwell stability control testing*" (2016). These standards were developed to provide guidance for OEMs and testing organizations applying the regulations UN/ECE Regulation No. 13-H, "*Uniform provisions concerning the approval of passenger cars with regard to braking*" and USA FMVSS 126 "*Federal Register Vol. 72, No. 66, April 6, 2007*."

These regulations specify the use of a steady-turning test called "slowly increasing steer" and a dynamic steering test called "sine with dwell" to evaluate the capability of a vehicle equipped with an electronic stability control (ESC) system to maintain stability and control at limit conditions. The UN/ECE regulation specifically encourages testing organizations to use simulation to evaluate variants of a particular vehicle model class to complement limited physical testing.

Both of these ISO standards contain the same introductory statement:

> "The main purpose of this International Standard is to provide a repeatable and discriminatory method for comparing simulation results to measured test data from a physical vehicle for a specific type of test."

In each case, a vehicle is physically tested with instrumentation using the specified procedure. The simulation tool is used to replicate the same vehicle undergoing the same procedure.

The steady-turning tests involve the measurement of steering wheel angle and a few motion outputs, obtained in either steady-state, or tightly defined slowing changing conditions (e.g., slowly increasing steer, applied with a robot controller). Thresholds are defined over the range of performance to determine whether the results of the simulation tool match repeated tests well enough to be "valid" for this type of test.

The sine-with-dwell tests involve a slowly increasing steer test, used to set up a series of dynamic robot steering tests called sine with dwell. For each sine-with-dwell test, a few critical times of interest are identified, and tolerances are specified. If values of output variables from the simulation tool agree with the variables obtained with physical measurement, then the tool is declared "valid" for this type of test.

Proprietary Validation Standards

Simulation of vehicle dynamics has been used in industry for decades and has grown to be an essential part of the development process.

The simulation tools of choice have shifted over time from in-house proprietary software to commercial software such as the VehicleSim products from Mechanical Simulation, and more generic multibody simulation tools such as Adams (from MSC.Software) and others.

Although OEMs have converged to using simulation tools from a common set of commercial products, there have not been generally accepted published validation methods. Researchers, engineers, and software developers have largely relied on "common sense" and "rules of thumb" for validating tools. To our knowledge (at Mechanical Simulation), most OEMs who use CarSim, TruckSim, and BikeSim apply their own in-house methods for validating new software tools and the datasets they obtain to represent specific vehicles. Most consider their methods to be proprietary, and to provide a competitive advantage that they do not wish to share.

In Europe, where multiple countries share responsibility for regulatory compliance, some OEMs were frustrated a few years ago when their in-house methods were not known or accepted by other organizations, who rejected their claims of a "valid" tool. This was a factor driving the ISO to develop and publish the validation standards ISO 19364 and ISO 19365 described earlier.

Even though ISO is continuing to create validation standards for other types of tests, the process is slow, and mainly limited to areas where existing standardized test methods might be used in regulations.

Validation by Comparison with other Software

Validation by comparing simulation results to measured test data involves two expensive tasks:

1. A physical vehicle must be obtained, instrumented, and operated according to the specific test procedures of interest. Measurements must often be processed to generate a report in a specified format.
2. The properties of the physical vehicle must be obtained by measurement or other means to obtain the inputs needed by the simulation tool. Environmental conditions (ground 3D geometry, friction, temperature, etc.) must also be considered.

Because of these expenses, validation of both a software tool and the methods used to obtain inputs for the tool is beyond the capabilities of Mechanical Simulation (and most software companies), and most users other than OEMs or research labs with test facilities.

Comparisons with Similar Software

A far more common form of validation is to test a software tool by using another software tool as a reference. This tests the calculation methods used by the software tool when given inputs that are also used for the reference simulation.

For example, Mechanical Simulation constantly tests new versions of each software product in comparison with past versions. This is done manually by developers when a new feature is added. We contrive tests in which we expect the new feature to have no effect on the simulation results and confirm that this is the case for a set of simulation tests. We also create some tests that are intended to demonstrate the influence of the new feature, with some expectation of what the differences should be. Because nearly all inputs for the models are consistent between versions, we know that observed differences are caused by differences in the modeling.

Existing simulation datasets are rerun via automation for new software revisions, to confirm that new versions match older versions except when improvements are made in which some differences are expected.

New users or potential users that have access to a "trusted" simulation tool will test a VehicleSim product by preparing inputs for a known vehicle and comparing results with a known tool. This task takes a little more effort because different simulation tools often have different types of input. It is often necessary to simulate a simplified version of the vehicle using parameters that are common to the different tools.

Comparisons with Simple Models and Hand Calculations

There have been several occasions where new users of CarSim or TruckSim compared simulation results for simple tests with results calculated from simple models as presented in technical papers, textbooks, or fundamental force balances.

For example, we have received several tech support questions based on comparisons of metrics from steady braking tests in CarSim with metrics calculated with formulas from the textbook by Tom Gillespie, "*Fundamentals of Vehicle Dynamics*," Society of Automotive Engineers, 1992. (Dr. Gillespie is a co-founder of Mechanical Simulation Corporation, now retired.) Load transfer and longitudinal acceleration metrics are close, but not identical. The questions are usually in the form: "The results from CarSim do not match the values we calculated from the textbook. Why is CarSim wrong?" (And of course, the answer is "The CarSim results are more realistic.") In the case of steady braking, there are factors in the detailed 3D model that are not included in the simple

analyses. One factor that most new users do not consider is that when braking without skidding, the wheel spin changes, causing a moment applied to the vehicle due to the product of the spin acceleration and the tire/wheel spin inertia. Other factors include aerodynamics and changes in pitch and C.G. location of the sprung mass due to the anti-pitch geometry of the suspensions.

Comparing Versions of VS Vehicle Models

This section describes the methods used routinely at Mechanical Simulation to compare results from new versions of a software product to results obtained with past versions. Various levels of agreement might be seen. The examples in this section involve simple maneuvers and continuous behavior. (More extreme conditions are discussed in the next section.)

Depending on what changes are made between versions, various levels of agreement are expected. The main method for rapidly identifying the amount of agreement between two versions is to generate results in a database running under the old version (e.g., a 2019.1 database) for all runs of interest, and then run the same simulations with the new version (e.g., 2020.0). It will often be necessary for the database to be updated to the new version. Make sure the database is locked. Then, for each run of interest, make a new run and compare the results:

1. Duplicate the Run Control dataset for the run of interest.
2. Choose a different color (click the box Set color in the upper-right corner of the screen and then choose a distinctly different color). For example, if the original vehicle color is blue, choose a light color not related to blue (e.g., bright yellow).
3. Click the Run button.
4. Check the box Overlay videos and plots with other runs and select the original run that was duplicated for an overlay.
5. Use the View button in the lower-right region of the Run Control screen to view the Echo file with final conditions for each run and compare the two Echo files in a text editor.
6. Click the Video + Plot button and view the overlaid videos and plots.

Compare State Variable Values in End Echo Files

The most vigorous comparison is to look for differences in the values of the model state variables (SVs) listed in the Echo file made at the end of the file, viewed using the View button as described above. Load both files into a text editor such as ConTEXT (the editor provided in VehicleSim products) and use the Tools option Compare to view differences in the two text files. Differences in the files are highlighted in yellow (Figure 1).

Expect to see the top portions of the files differ due to different titles. Changes might be seen due to text describing parameters (Figure 1), or because new parameters are shown, or factors other than the numerical values of model variables.

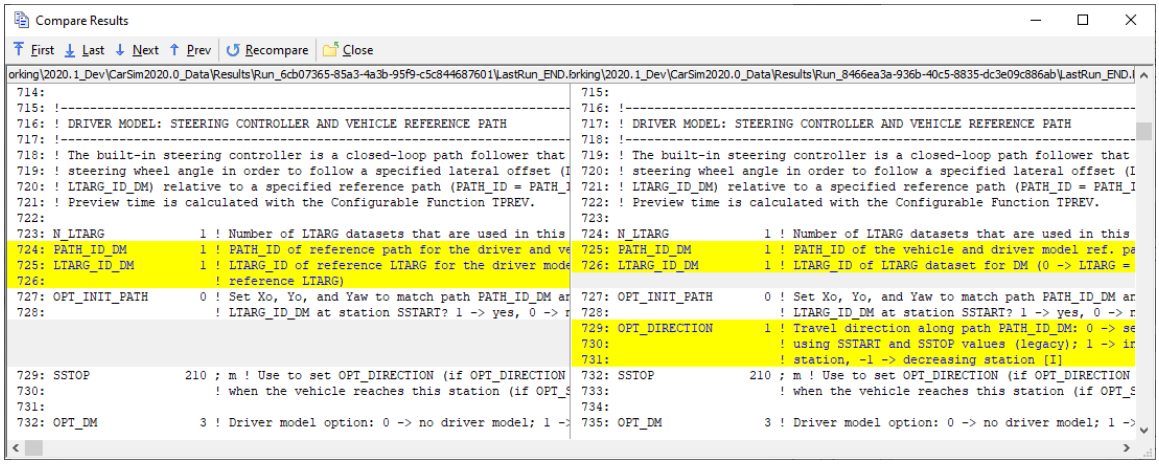

Figure 1. Differences in End Echo files due to changes in comment text.

The section to inspect is the end part of the files, which lists all model SVs and their values when the simulation run ended. For example, Figure 2 shows yellow regions for the SVs named `SV_SO_DIST` and `SV_STR_DM_OLD`. In this example, the differences are due to minor changes in the labels; the numerical values shown for the SVs match completely for the printed resolution (9 and 10 digits for the two cases).

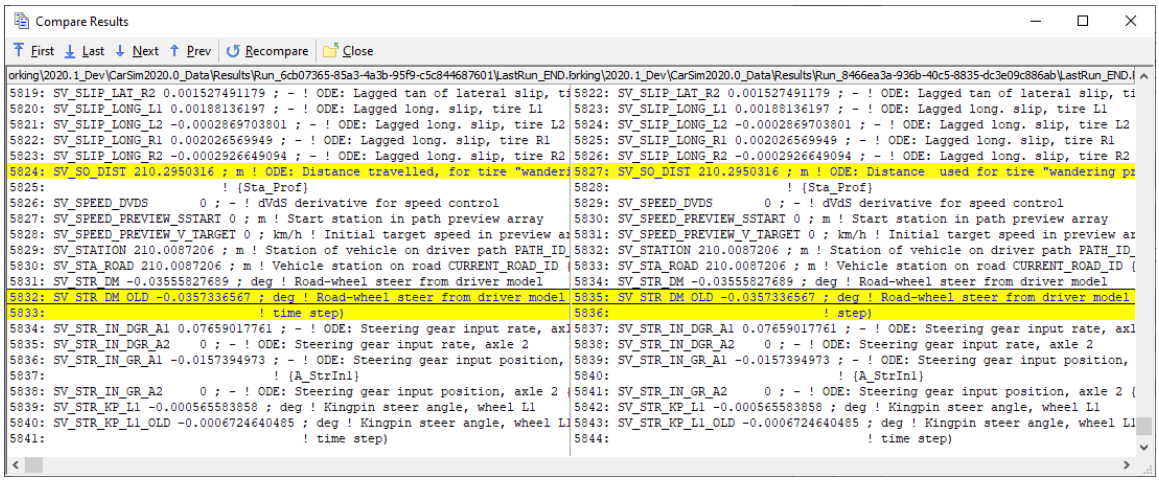

Figure 2. Look for differences in the state variables (SVs), listed at the end of the End Echo files.

This type of comparison is made routinely at Mechanical Simulation via automation. It has the advantage of being easy to apply and is probably the fastest means for determining that two versions have "perfect agreement." However, if the agreement is not "perfect" to the precision used for writing the numerical values, it is not the best way to determine if results are "close enough."

This is an extremely rigorous comparison. Consider that the solver has tens of thousands of variables that are typically calculated 2000 times for each simulated second. If any changes exist with any state variables, anywhere during the simulation, they will affect the values that are written at the end of the simulation in the End Echo file.

No Flicker in Video Overlays

Next, consider overlaid videos. If videos overlay perfectly, the video will not show any evidence of the new run because there will be no offset or rotation between the 3D objects as located in the two simulations. If the original was blue and the new color is yellow, the vehicle will appear blue throughout the overlay video (Figure 3). In this case both simulations are very close in calculating the motion variables that affect the animations.

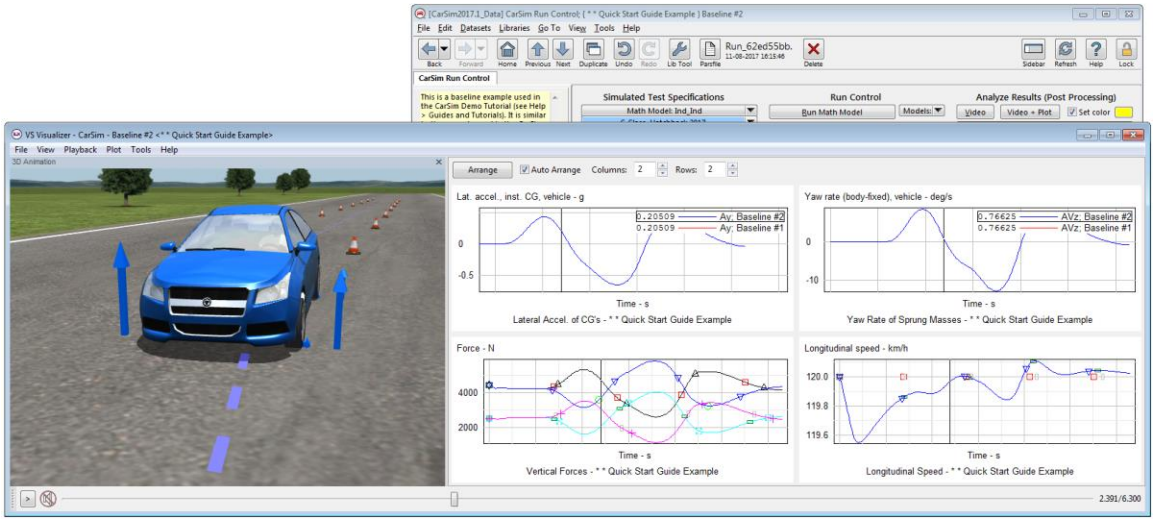

Figure 3. Perfect agreement (no flicker, same values in digital displays).

The plot windows should show plotted data from both runs, with different colors used for the different variables. For example, the legend for the upper-left plot window (Figure 3) shows one label in blue and the other red. Because the results overlay so closely, only one color is visible for the actual plots (blue). To confirm that both datasets do exist, view any of the plots and confirm that results from both runs overlay. With a plot window active, type ‘v’ to view digital values in the text legend. You should see that values agree from the two versions for all digits, or sometimes all but the last.

Flicker in Video Overlays

Sometimes VS Visualizer shows a flickering between the two colors associated with the vehicle (Figure 4). The flickering indicates that the surfaces for the vehicles in the two simulations do not match within the resolution of the virtual video camera in VS Visualizer. When both colors are visible in patches over the shape, the differences in location are typically a millimeter or less. This small discrepancy is beyond the resolution of most measurement systems but is visible in simulations.

The plots associated with the two simulations overlay such that no differences are visible with normal scaling. Although multiple lines are shown with different colors, they overlay so closely that only one color is visible (blue in the top two plots). When text values are shown, values agree with 4 or 5 digits (typically a fraction of a percent of the peak values shown).

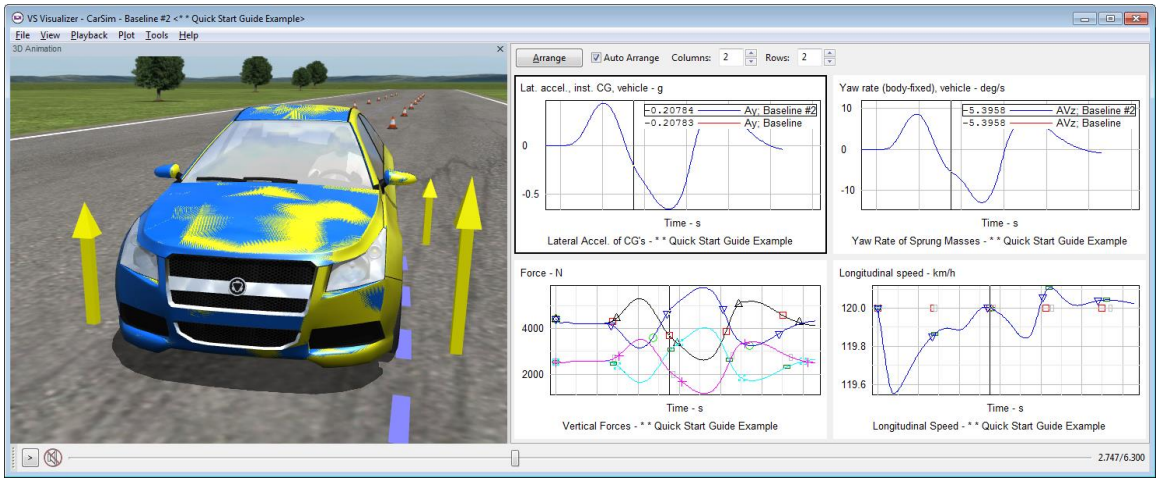

Figure 4. Overlay with video showing "flickering" and plots showing very close agreement.

This example (Figure 4) compares results between CarSim 2017.1 and a release candidate for 2018.0. Many changes were made internally in the creation of a single modular VS Solver that handles all vehicle configurations. Although no differences can be seen in the plots with normal viewing, some differences exist beyond the first 4 or 5 digits.

Small Differences in Plots

Figure 5 shows a comparison in which the general behavior matches, but small differences in plots of interest are visible. In these cases, the videos will show nearly identical motions. However, rather than the flickering seen when locations are at the resolution limits, the vehicle colors seen in the video overlays show the two simulated vehicles have distinctly different positions. In this example, the left side shows blue, and the right side shows yellow. This indicates that on the left side of the view, the blue vehicle is closer to the camera, while on the right side, the yellow version is closer.

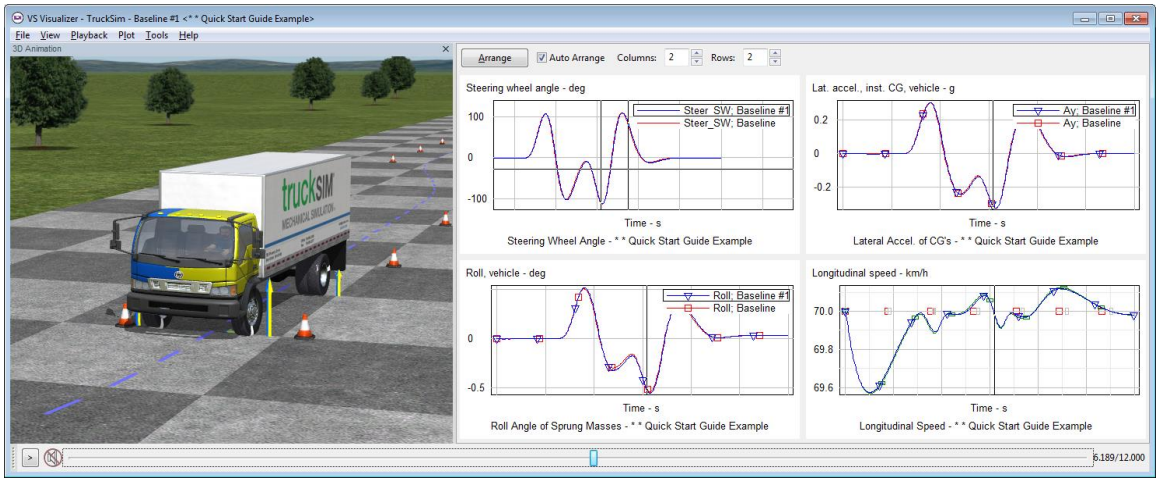

Figure 5. Overlay showing small but visible differences in plots.

The plots are qualitatively similar. As in the other examples, the top two plots involve overlays of two datasets shown with different colors. However, in this example, both colors (blue and red) can

be seen. Although the plot match closely, the agreement is not at the same level as in the examples shown in Figure 3 and Figure 4.

This example shows the Quick Start Guide double lane change from TruckSim, comparing version 2017.1 to a candidate release for 2018.0. In the new version, the steering geometry for TruckSim was improved to include more 3D detail in the movements and torques related to an inclined kingpin axis, including secondary effects of vertical load causing compliance steer, gyroscopic effects, and other moments related to physical movement of the tire center of contact.

Qualitative Agreement

The results shown so far are made in an ideal sense of using the same inputs to different versions of a specific software tool. When comparing results from different simulation tools, there may be significant differences because the model descriptions are not equivalent. Properties in one tool might not exist in the other. Conventions used for characterizing properties might differ, and a user who is not familiar with a tool might provide information with the wrong units, wrong sign convention, or fundamentally incompatible with the requirements.

Figure 6 compares results from two CarSim runs in which two unrelated vehicle datasets are used for the same simulated test. (One is the C-class hatchback, the other is a D-class SUV.) Although the vehicle properties are different, both simulations use the same test conditions (double lane change at 120 km/h on a flat surface, using close-loop controllers for speed and path), and the animation shapes (those for the C-class hatchback). Because the two simulations use the same test conditions, the general shapes of plots are similar in timing and scale.

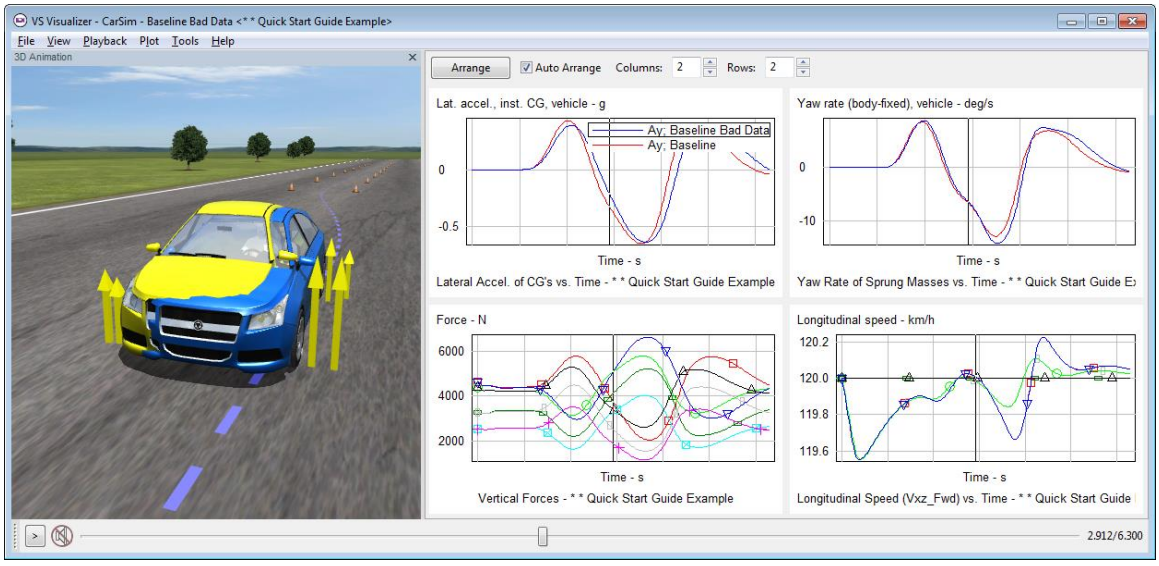

Figure 6. Comparison of results using different vehicle datasets.

Although all results are feasible for a C or D class vehicle performing the maneuver, the differences show the effects of the alternative vehicle characteristics.

Summary of Comparison Methods

The comparison methods described in the above subsections are summarized in Table 1, going from closest agreement to least agreement.

Table 1. Summary of comparison methods.

Comparison Method	Criteria	Notes
Compare written SV values in text file	No difference in text for numbers	Suitable for automation, this is the most stringent method
No flicker in video overlays	No flickering in video	Also stringent, good for interactive viewing
Flicker in video overlays	Motions match, flicker is visible, plots match visibly	Still close agreement, good enough for most cases
Small differences in plots	Video nearly matches, plots nearly match	Still close, good to have reason why there are differences
Qualitative agreement	Video and plots are similar	Differences are significant

Sources of Disagreement Between Versions

The VS Solvers used for CarSim, TruckSim, and BikeSim have a few hundred SVs, used along with hundreds or thousands of parameters and table values that define the state of the model at any instant of time. Some of the state variables are defined with ordinary differential equations (ODEs) that are solved each time step by numerically integrating the derivatives of the ODE SVs. The models also include hundreds to thousands of additional output variables of interest that are calculated from the SVs. All the internal variables are either integers or 64-bit floating point "doubles."

If the modeling assumptions are the same in the two versions but changes are made in the code organization or compiler options, the only expected differences are due to round-off of 64-bit arithmetic. (Compilers optimize the sequence of calculations, so differences due to round-off effects are always expected when the compiler is adjusted.) A 64-bit double has about 16 significant decimal digits, so we expect round-off differences to be unnoticeable in most cases.

Note C code for multibody kinematics and dynamics are generated at Mechanical Simulation by a symbolic multibody program called VS Lisp, based on fundamental modeling definitions involving relative movements of rigid bodies subject to kinematical constraints. This provides consistency for the 3D multibody equations; the machine-generated code is very consistent, regardless of how the solver architecture evolves over versions.

Simulations with Continuous Behavior

The comparisons shown in the previous section illustrate the range of agreement that can be seen and expected when looking only at the software tool if the simulations do not involve limit conditions or discontinuities such as gear shifts, controller interventions (ABS, ESC, etc.), tires lifting off the ground, skidding, etc.

The VehicleSim solvers are coded with a modular architecture that allows a single VS Solver (e.g., `trucksim_32.dll`) to handle all vehicle configurations involving trailers, suspension types, and options such as frame twist. The modular architecture was introduced for CarSim and TruckSim 2018.0; for BikeSim, the modular architecture was introduced in version 2020.0. For single-unit configurations (no trailers), there were no model changes needed due to only modularity. When comparing modular versions to earlier versions, you will usually see at least "video flicker" agreement. TruckSim 2018.0 includes model improvements in the 3D steering geometry, so comparisons with versions 2017.1 and earlier typically show small differences in plots, as seen earlier in Figure 5 (page 9).

The hitch model used for CarSim and TruckSim vehicles with trailers now includes some compliance. The modeling assumption of compliance provides some stability and computational speed advantages in how the semitrailer modules are activated. (Past versions assumed a rigid connection between hitch points in a leading and trailing unit.) Therefore, small differences will usually be seen when comparing CarSim and TruckSim combination vehicles for versions 2018.0 and 2017.1.

Divergent Results

Many simulations include discontinuous events, such as a gear shift, a controller intervention, a tire lifting off, etc.

For example, recent versions of CarSim have included a simulation showing how the model can be extended using custom forces and reference points together with VS Commands to add outriggers of the sort sometimes used in testing limit rollover conditions. When comparing results from versions 2018.0 to 2017.1, close agreement is seen early in the simulation, up to the time when the left-rear tire lifts off the ground (Figure 7). After that time, the simulation shows a violent reaction as the outriggers touch the ground and the vehicle rebounds in roll to the other direction. Seconds later, the positions predicted by the two versions have diverged (Figure 8).

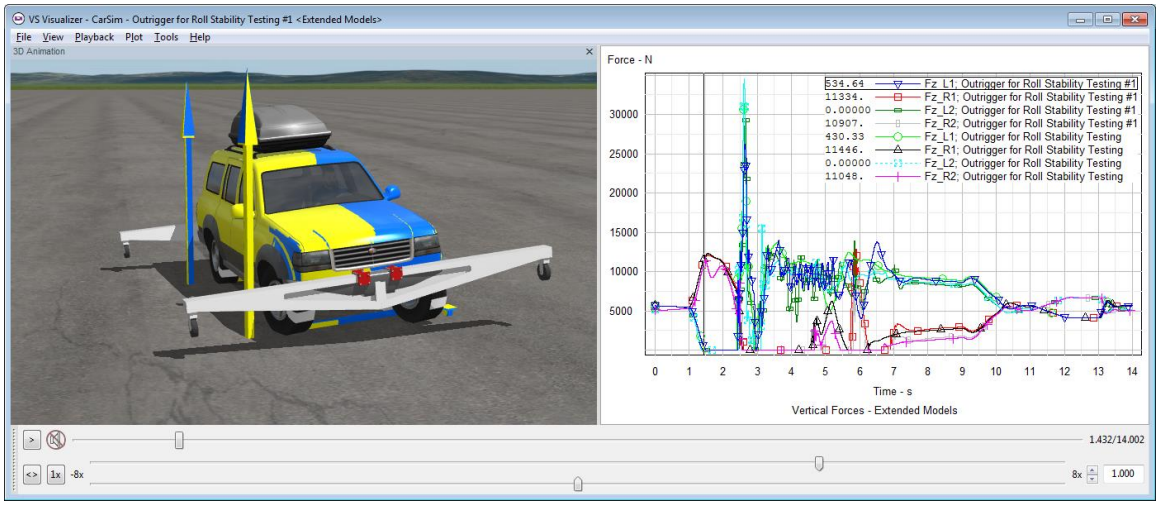

Figure 7. Comparison of outrigger example in CarSim when a tire first lifts off.

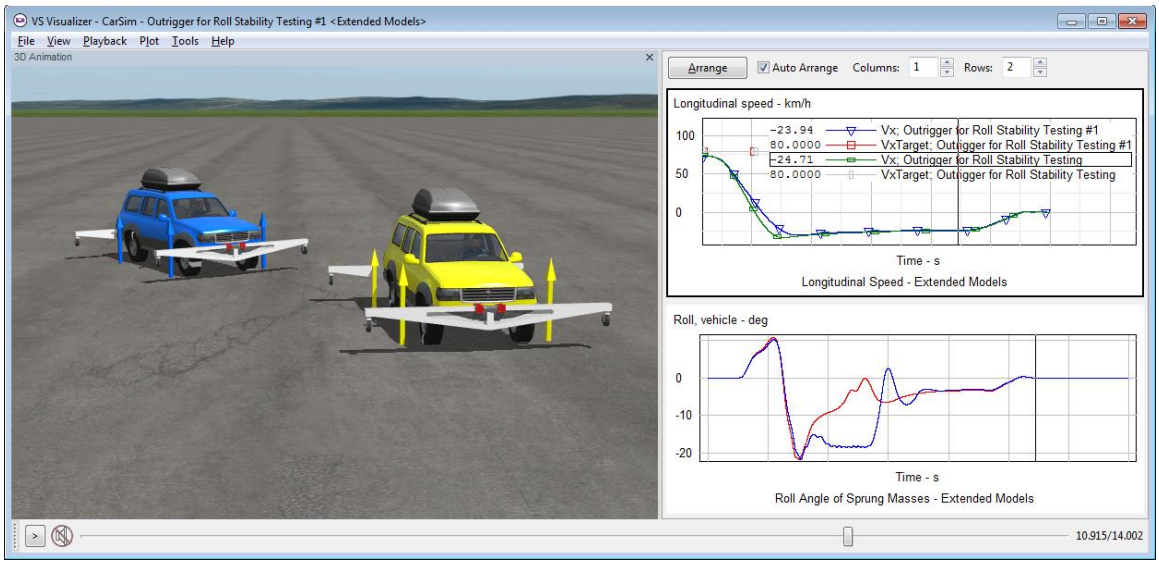

Figure 8.Divergent results near the end of the simulation for the outrigger example in CarSim.

At different times in the outrigger simulations, all tires of the vehicle lift off the ground. When on the ground, friction limits are sometimes reached in lateral force generation. Also, the outriggers generate large vertical force pulses when specified points on the outrigger hit the ground.

The two models agree closely in the first 1.48s of the simulation; however, the conditions where the first tire lifts off are not identical. From that time on, the models diverge slightly, until the next discontinuous event occurs (another tire lifts off). At that time, the simulations diverge further. This simulation has discontinuities involving each tire lifting off, and each outrigger contact point hitting the pavement and generating a large force pulse.

Source of Divergent Behavior

Given that the SVs of a multibody vehicle model are calculated by numerically integrating derivatives from ODEs, the results are subject to integration error along with round-off involving numerical calculations. However, the nature of a vehicle design is that the ODEs are somewhat self-correcting. For example, if a calculated spring deflection is slightly smaller than it should be, the calculated spring force acts to move the bodies in the directions needed to correct the error. If a vehicle simulation runs for a very long time, the springs and dampers in the tires and suspensions will maintain the correct conditions.

Drift can occur for motions that have no built-in restoring force. For example, there are no built-in forces or moments applied to a vehicle based on its global location; if two simulations are being compared and one vehicle is ahead of the other by 1 meter, there is no built-in feedback that would cause the gap to change over time. Similarly, if steering control is open-loop (as with a robot), and the vehicles from different simulations have different yaw angles, there is nothing to bring the yaw angles into agreement.

The outrigger example presented in the previous subsection has a closed-loop speed controller at the start of the run, followed after 1.0 s by coasting in neutral and open-loop steering during the rest of the test. Therefore, if the two vehicles being compared reach different positions or yaw angles, there are no actions (forces or moments) that would bring them together. Once they have

diverged at about 1.48s (when the first tire lift-off occurs), they will follow different paths. New discontinuities (tires touch back down, outrigger contact with ground, etc.) occur at slightly different times, cause more divergence.

Any action that affects position, orientation, or speed can cause a divergence that has no built-in restoring effect.

Discontinuous Changes

Although the multibody equations are continuous, the models include some discontinuous options where a small change causes a discrete change in other variables. This occurs with tires losing contact with the ground (all forces and moments for the tire go unconditionally to zero). It also occurs with on-off controllers, such as an ABS controller that is either 0 or 1. Regardless of the size of the change that triggers the on-off control to change, the effect is discrete and will cause a significant change in behavior immediately. If two models differ by a very small amount, but one triggers the controller to change one time step before the other, then the results will begin to diverge.

In addition to the built-in ABS controllers, the powertrain has clutches and gears that can produce discrete changes that can introduce a divergence between two models that are similar but not identical.

Iterative Calculations

Most of the equations in a VehicleSim math model are explicit assignment statements of the form:

x = math expression of known variables

where x is variable in the math model. This type of statement has no ambiguity; if all variables on the right-hand side of the '=' sign are given a certain set of values, then the same result for x will always be obtained.

However, some of the calculations involve implicit conditions, where the math expression on the right-hand side is dependent somehow on the value of the unknown variable x.

For example, given a known position (X-Y-Z) of a wheel center in 3D space, and its orientation (three sequential rotation angles), the tire radius depends on the ground elevation where the tire contacts the ground. However, the ground elevation usually depends on X-Y coordinates, which in turn depend on the radius of the tire if the wheel is not perpendicular to the slope of the ground at the point of contact.

If the intent of the math model is to determine variables such as tire radius to the best possible precision, an iterative solution method would be used in which a radius is assumed, the ground X-Y coordinates are calculated, and a new value for the radius is calculated. If the new value for the radius is not within some tight tolerance of the assumed value, the calculations would be repeated using the new value of radius to find the X-Y coordinates. The process would be repeated until the new and old values of the radius match within the tolerance.

The VehicleSim math models are not intended to provide the absolute best value of all internal variables at every time step; they are intended to run fast, in support of real-time applications, and to give results that are "close enough" (e.g., 6 digits of agreement, rather than 14 digits).

Calculations such as a new tire radius are typically handled with one iteration, using the value from the previous time step as the "old" value, and using the new value for the current time step.

The VehicleSim models include several types of calculations that are inherently iterative, including:

1. The calculation of tire radius (described above) on a surface that is not uniformly flat.
2. The calculation of station along a path given global X and Y coordinates.

Iterative methods are either applied only once (e.g., the tire radius), or are repeated until an internal error tolerance is achieved (e.g., case 2 above). In those cases, a very small difference in state variable values can result in a larger difference related to the internal error tolerance used for an iterative calculation.

Round-off for Global X and Y Coordinates

All floating-point calculations within a VS Solver are made with 64-bit variables. However, the legacy ERD file format used for output files employs a 32-bit binary format. Most output variables of interest are represented well with 32-bit resolution (6 significant digits), such as forces, moments, velocities, accelerations, angles, and relative displacements. Given that a file with 32-bit format is half the size of one with 64-bit format, there are advantages to using the smaller size. For this reason, the newer VS file format also supports 32-bit binary output files.

However, a few output variables may require more digits for acceptable precision:

1. GPS latitude and longitude are scaled to cover the entire surface of the world with angles ranging from ±90° latitude and ±180° longitude. In a typical simulation, the first 4 or 5 digits remain constant. Round-off is typically about 0.1 m if the angles are represented with 32 bits.
2. Global X and Y coordinates are normally set to be 0 at the start of a simulation. If the simulation covers long distances, the precision limits will be visible in animations as flickering. For example, with values near 10 km, a 1-mm change in position is the difference between 10000.001 and 10000.002. This requires 8-digit precision, which is not possible with 32-bit binaries.

When GPS or global X and Y coordinates are needed with more precision, the output file format should be set to 64-bit VS (`OPT_VS_FILETYPE` = 2).

Evaluating New Vehicle Datasets

The validity of a simulation depends on both the built-in capabilities of the software tool and the input data used to describe a specific vehicle of interest. Assuming that the validity of the tool is acceptable, there is still a need to build a dataset for a particular vehicle that will support the capability of the tool to provide valid results.

Most of the example simulations provided in CarSim, TruckSim, and BikeSim show interesting scenarios that involve combinations of closed-loop controls, open-loop controls, on-board controllers (e.g., ABS, ESC), possibly 3D ground surfaces with curbs, and possibly other moving objects. These types of scenarios are not suited for performing simple validations to determine if a new vehicle dataset is representing a specific vehicle of interest.

When assembling data for a new vehicle, several simple simulations should be made to confirm that the vehicle being simulated has the intended properties. CarSim and TruckSim include example simulations in the category (submenu) Validation of Vehicle Data. The examples use the same vehicle dataset that is used for the Quick Start Guide. All involve running the vehicle on a flat and level surface using only open-loop controls.

Vehicle at Rest

The example simulation with the title Validation of Vehicle Data > At Rest has the vehicle at zero speed, located on a flat and level surface, with zero steer, zero braking control, and the powertrain in neutral.

Besides showing all the parameters and tables used to represent the vehicle, the Echo file includes some calculated properties, such as the overall mass (with payloads, if applicable) and CG location (Figure 9).

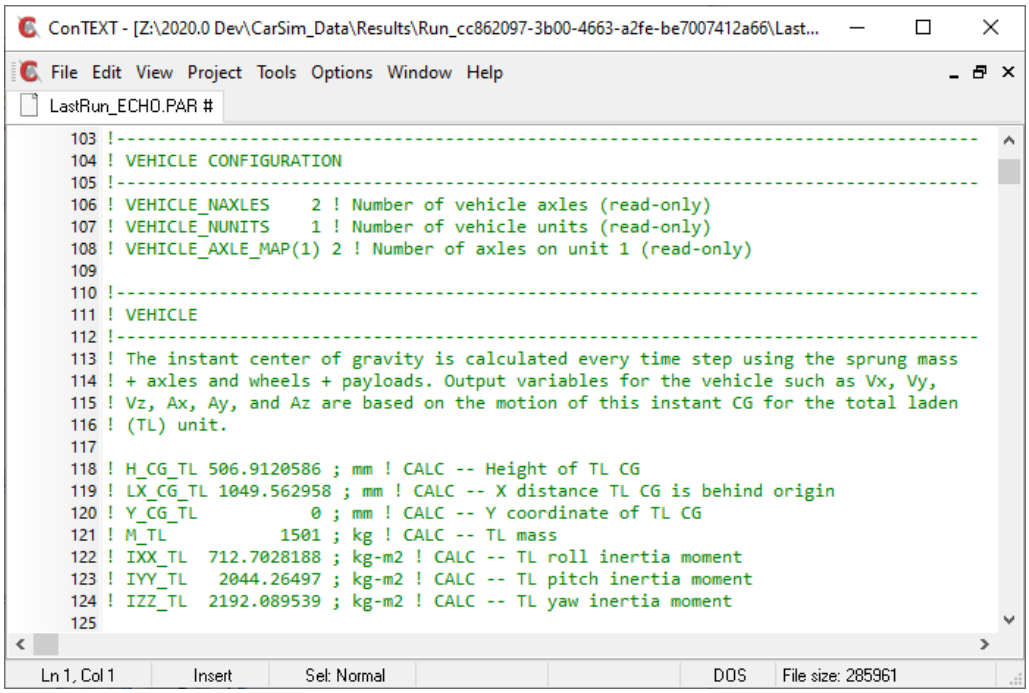

Figure 9. Total vehicle mass and C.G. location as shown in Echo file.

When information is provided in CarSim for a known vehicle, the first check is to see if the total mass and static ground loads match those of the known vehicle. For the CarSim C-Class hatchback (the vehicle used in the Quick Start Guide examples), the total mass is 1501 kg (line 121).

Differences between initialization and true equilibrium

CarSim and TruckSim perform a simple static initialization before starting. When the simulation starts, some adjustment will usually occur in the full 3D multibody model. For example, Figure 10 shows the small adjustments that take place when the vehicle moves from the approximate equilibrium into true equilibrium at T = 2s.

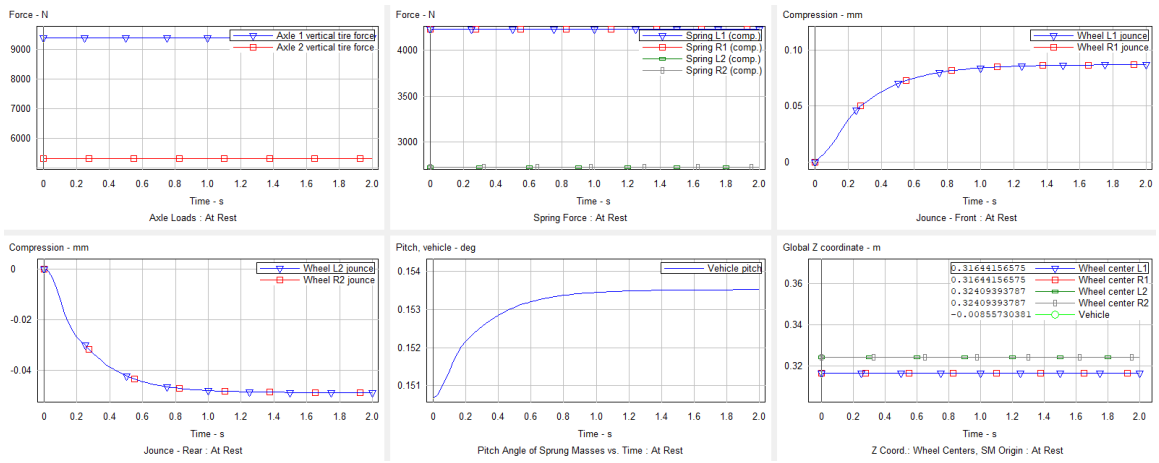

Figure 10. Time histories of movements and forces for vehicle at rest, values shown at T=2s.

Notice that the pitch angle is never zero. It starts at about 0.151° at T = 0 and reaches a value of about 0.1535° at T = 2s. The small positive pitch is due to a combination of three factors:

1. the sprung mass C.G. is forward, causing more load on the front axle;
2. the design heights of the wheel centers (heights of wheel centers when the sprung mass has zero pitch and roll) are the same for all wheels; and
3. the same tire is used for all four wheels.

With more load on the front axle, the front tires compress more than the rear tires, as confirmed by the values of the wheel center heights. The last plot in Figure 10 shows the height for a front wheel center is 0.316m; for a rear wheel center it is 0.324m.

The pitch and roll angles are based on the axes of the sprung mass coordinate system, used to define directions and locations of points of interest. The mass and inertia properties given on a Sprung Mass screen define the *Design Load* condition in CarSim and TruckSim, which is nearly always not a perfect equilibrium condition. For more information about the design load, please see the Technical Memo: *The Design Load Condition in CarSim and TruckSim*.

Suspension jounce and Z coordinates of vehicle parts

Suspension motions are characterized in part by kinematical tests in which motions of the wheel knuckle are measured as jounce changes. These include longitudinal motion due to jounce, lateral motion due to jounce, toe change due to jounce, camber change due to jounce, and pitch change (also called “dive”) due to jounce. CarSim and TruckSim support several options for defining where jounce is zero. The plots in Figure 10 were made for a vehicle in which jounce was defined as zero at the design load condition. Other options have jounce at a non-zero value specified explicitly or obtained using spring force/deflection data.

When viewing plots of jounce in the “At Rest” condition, confirm that the values shown are consistent with the expectation of what jounce should be in static equilibrium.

The vertical locations of the vehicle sprung mass and wheel centers are also shown in the plots in Figure 10. Locations are specified for most points of interest using the coordinate system of the sprung mass. The origin of this coordinate system can be located anywhere that is convenient.

However, it is essential that all coordinates are defined using the same location. All example datasets provided with CarSim and TruckSim have the origin of the sprung mass coordinate system for the lead unit located longitudinally at centers of the front wheels for a lead unit, or at the hitch point for a trailer. Laterally, the origin is located at the mid-point of the lateral positions of the wheel centers in a design condition. Vertically, it is located near the ground. Given that the vehicle may be simulated with different tires and/or loading conditions, the vertical location of the sprung mass (and therefore, the origin of its coordinate system) will not be the same under all conditions.

Relationship between suspension jounce and spring compression

When specifying properties of the ride springs and dampers in a suspension, there are several options available in CarSim and TruckSim. The internal model for a generic/independent suspension uses forces applied at the wheel center based on suspension jounce changes. If the properties of a physical vehicle are measured in a K&C test rig using measured changes in vertical load at the wheel center or ground along with measured changes in jounce, then that data may be used directly in the model. In this case, the ratio between suspension jounce and spring compression (as measured) is 1.0.

On the other hand, if the properties of a spring or damper are obtained by a separate testing rig, then the difference between suspension jounce and compression of the component must be specified. This is done with a potentially nonlinear table relating suspension jounce to component deflection.

The Echo file shows multiple calculated loads and compressions, shown as with descriptions that begin with "`CALC -`" to indicate that they are not input parameters, but are calculated from input parameters (Figure 11). The keywords and values are shown as comments because they cannot be set directly.

In this example, the static ground load for the front axle is 9410.7 N for the vehicle as laden (`FZA_L`, line 268), and the same for the ground load for the unladen condition (`FZA_UL`, line 269). In this example, no payloads were added to the vehicle, so the laden and unladen values are identical.

Other calculated values shown in the Echo file include the spring design force (`FS_STATIC`) and compression (`CMP_DESIGN`), which consider the suspension load (`FSA_L` and `FSA_DESIGN`) and the mechanical advantage of the suspension that related suspension jounce to spring compression.

Notice that the calculated design suspension load for the front axle is 8110.4 (line 262) is smaller than the sum of the spring forces (4228.6 + 4228.6 = 8457.2). The front suspension for this vehicle has a ratio of the spring compression to suspension travel set to 0. 0.959. This means that for suspension jounce of 10mm, the spring compression will only be 9.59mm. It also means that for each 10N of spring force generated, 9.59 N is applied in the vertical direction between the sprung and unsprung mass, with another component of the force vector being applied in a direction where no movement is possible. If the ratio is the same throughout the range of motion (which is the case for this example), the sum of the spring forces is the design suspension load divided by 0.959.

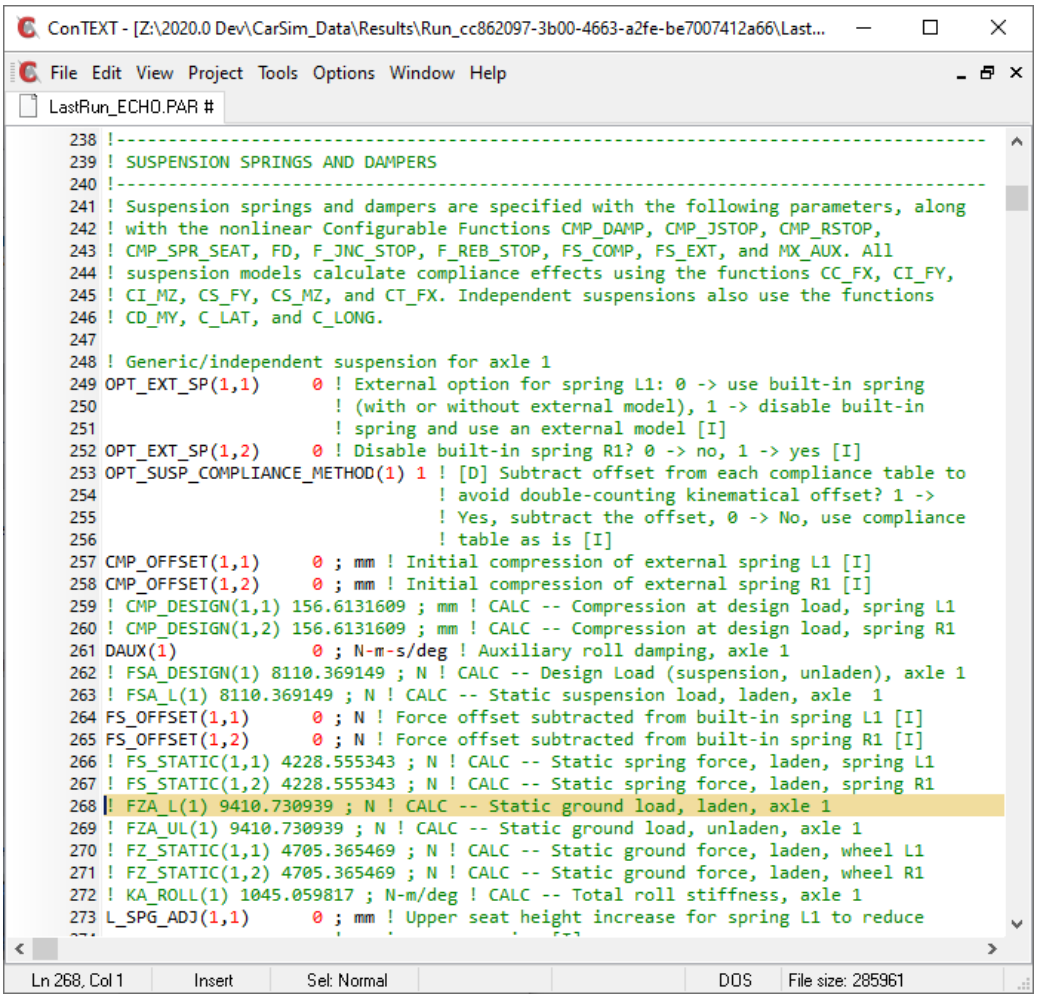

Figure 11. Part of Echo file showing static loads for ground and suspensions.

Considerations for TruckSim simulations at rest

TruckSim vehicles will typically show more transient adjustments in the "At Rest" test then the CarSim example that has been discussed. The simple initialization calculations are based on a symmetric vehicle with two axles. The assumptions in the initialization match the typical CarSim vehicle better than many of the TruckSim vehicles.

Most heavy trucks have an asymmetric steering system, in which one of the front wheels is connected directly to the steering system and the other front wheel is connected to the first wheel with a tie rod that has some compliance. Overall, the wheel connected by the tie rod has more overall steer compliance than the wheel connected directly to the steering system. The wheels are typically subjected to a moment about the inclined kingpin steering axis due to the vertical loads, and therefore the initial compliant steer effects are different on the two sides. The steering in turn causes small changes in vertical positions of the wheel centers, resulting in a small roll of the sprung mass.

If a lead vehicle unit has more than two axles, or if a semitrailer has more than one axle, then the true equilibrium condition will involve load sharing. The simple initialization calculates static suspension loads based on an assumption of perfect load sharing between multiple axles in a group.

These assumptions are adequate for getting the vehicle near the equilibrium condition, but some adjustment will occur.

Figure 12 shows plots for a 5-axle unladen tractor in an “At Rest” simulation. The pitch transient lasts about 4s and is much more oscillatory than the CarSim example shown in Figure 10.

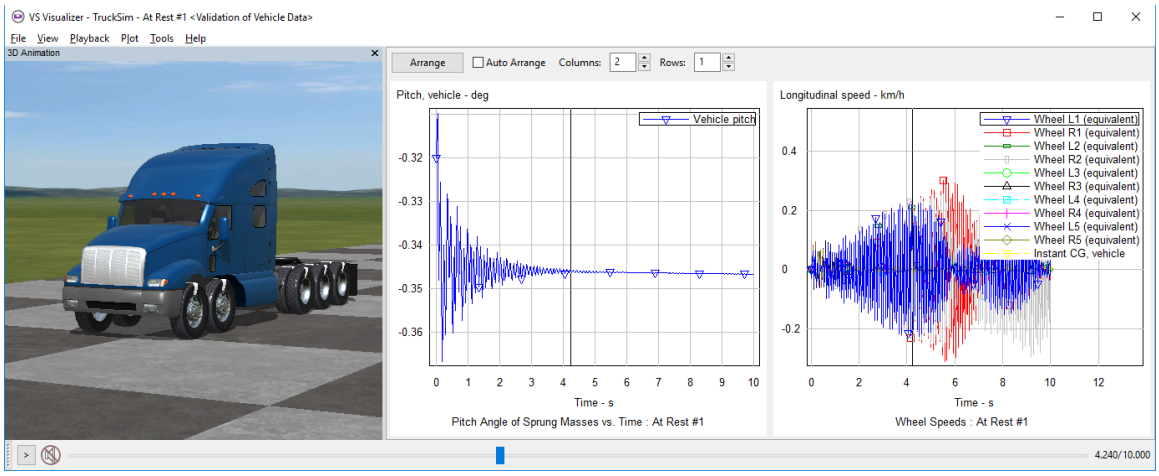

Figure 12. Adjustments in pitch for a 5-axle unladen tractor, with wheel spin oscillations.

The figure also shows a plot of tire longitudinal speed (spin, multiplied by rolling radius to obtain units of translational speed). Oscillations like this can occur when the vehicle is at rest and the wheels are free to roll, with no braking or connection to the powertrain. With the automatic scaling of the VS Visualizer plotter, the oscillations fill the vertical area of the plot and visually appear to be severe. However, the magnitudes shown are small, with maximum amplitudes of about 0.3 km/h. Motions of the tire via the white stripes or hub edges are not visible in the video, other than a “shimmering” of the tread texture that is sometimes visible when the view is zoomed into a tire.

These vibrations do not exist when the brakes are applied or when the vehicle is moving. However, it can be noticeable when the model is used in a driving simulator or in real-time hardware in the loop (HIL) testing scenarios where the vehicle will be at rest for some time at zero speed. In these cases, there is a simple fix: a wheel bearing friction parameter exists (keyword = `MY_FRICTION`) that is set on the Brake System screens. The default value is 0.5 N-m. When heavy truck tire/wheel assemblies have large moments of inertia, a larger value can be used to eliminate vibrations (Figure 13). In this case, the friction levels were set to 2 N-m for the wheels on the first two axles, and 4 N-m for the dual wheels on the other axles; the wheel spin oscillations no longer occur.

Slowly Increasing Speed

CarSim and TruckSim are used for many applications in which the powertrain behavior is not of great interest (handling, braking, intervention of controllers that mainly use the brakes, etc.). Engineers working in these areas do not always have access to detailed data describing the powertrain for the vehicle of interest, and might use powertrain data for a similar vehicle.

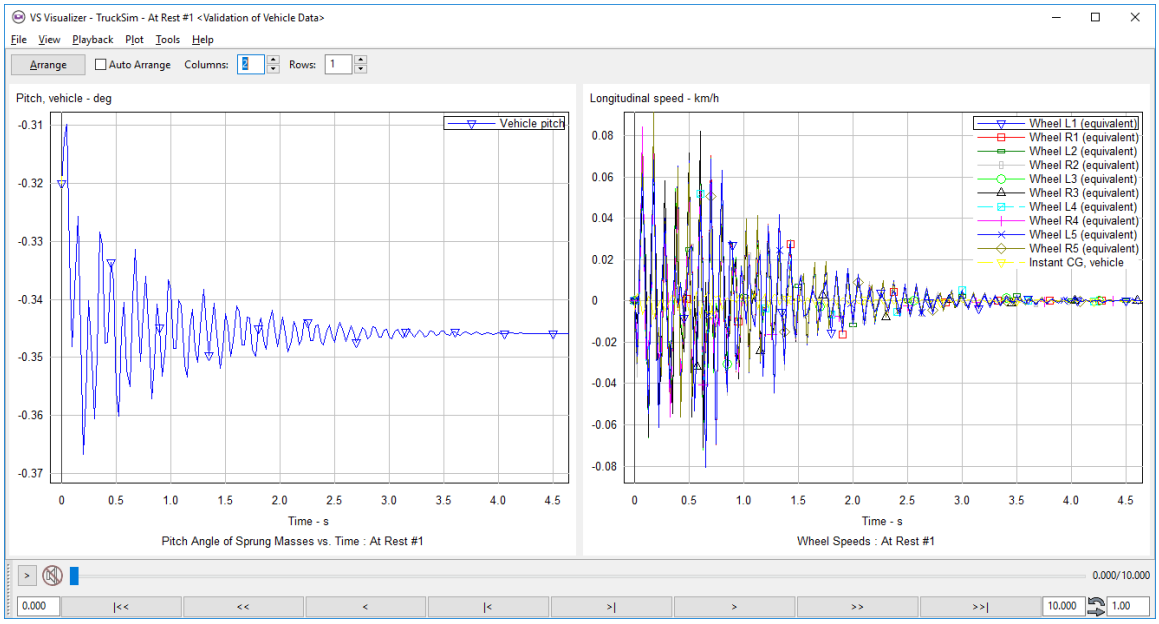

Figure 13. Transient results for the 5-axle tractor with higher bearing friction.

If the intent is to have a powertrain in the model that matches the basic attributes (front, rear, or all-wheel drive, number of gears, basic power capability, etc.), a slowly increasing speed test can be helpful to determine if the behavior meets expectations. Figure 14 shows results from a CarSim simulation dataset with the title Validation of Vehicle Data > Slowly Increasing Speed. This simulation is run on a flat, level surface, with the gear shifting set for automatic mode, with no braking or steering. The throttle is set to ramp from zero to 0.2 over a time of 5s and remain at 0.2 for the duration of the run. The simulation stops (via a VS Event) when the speed reaches 120 km/h.

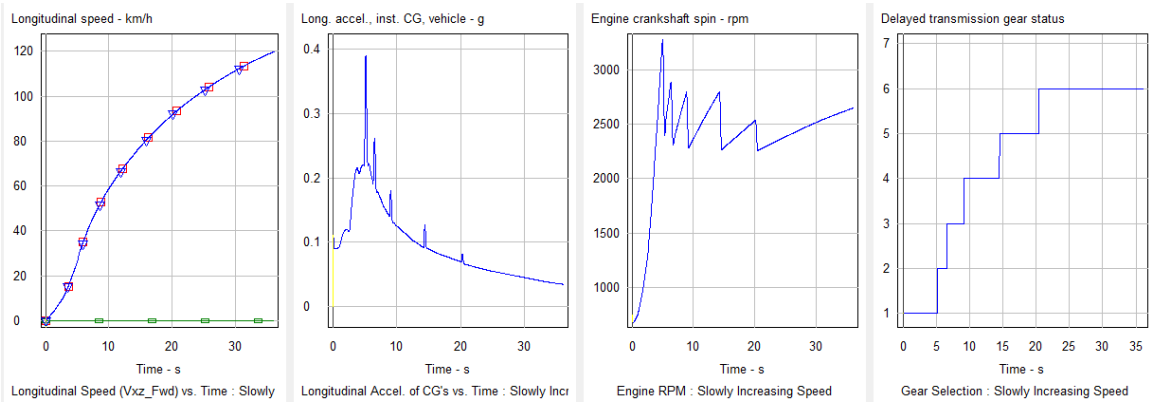

Figure 14. Plots for a vehicle accelerating slowly (0.1 throttle) to 120 km/h.

The plots show that the observed behavior is as might be expected, with acceleration ranging from 0.2g to 0.1g (except for spikes when gears shifts occur). The shifts make use of all six of the gears available in the example vehicle.

Engineers in a company using CarSim or TruckSim often share datasets. Although the VS Math Models cover all kinds of simulation environments and have parameters and tables to represent all relevant vehicle properties, engineers using the software for a specific application will typically be

concerned with obtaining good datasets for the vehicle properties that are the most relevant to their applications. For example, a group involved with braking and stability performance will need good data for suspensions, overall vehicle size and weight properties, the brake system, and possibly the steering system. If most of the simulations involve constant speed or braking conditions, then they might not be concerned about having representative powertrain data. If the datasets are shared with another group simulating conditions where acceleration or interactions with engine dynamics are important, the relevant properties might not be represented properly. For example, we (Mechanical Simulation) have seen problems through tech support in which shifting schedules are not matched to a vehicle being simulated, such that the higher gears are never used.

If powertrain dynamics are not critical for the simulations of interest, and you do not have powertrain data for the vehicle of interest, consider using the minimum powertrain option (speed controller only) in CarSim and TruckSim. With this option, the powertrain is specified with a maximum power parameter and a ratio indicating which axle(s) receive power. In CarSim, a rear-drive ratio is set to 0 (front-wheel drive, Figure 15), 1 (rear-wheel drive), or 0.5 (all-wheels). (TruckSim has ratios for the number of axles on the lead unit.)

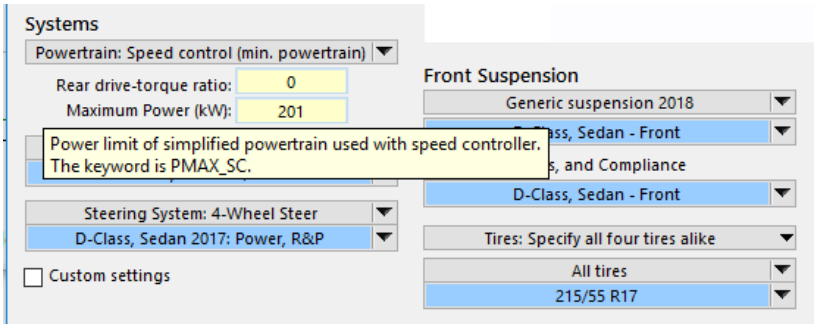

Figure 15. Minimum powertrain option.

If you do not have powertrain data, there are two advantages for using the minimum powertrain for speed control:

1. It requires only two numbers, and the math model will not generate misleading behavior for a more detailed model that is not described with the same quality as other properties.
2. If the datasets are shared with other users in your company, there will be no misunderstanding that powertrain data for the specific vehicle of interest were included.

Slowly Increasing Steer

Steering during normal city and highway driving causes vehicle motions that are affected by the overall size and weight parameters, suspension properties, steering system properties, and tire properties. The slowly increasing steer test (see the dataset with the title Validation of Vehicle Data > Slowly Increasing Speed) has the vehicle going on a flat, level surface at constant speed for 2s, at which time the steering wheel angle is slowly increased with time until a specified lateral acceleration is reached. Figure 16 shows the example for CarSim, in which the steer is increased at 2°/s at a speed of 80 km/h.

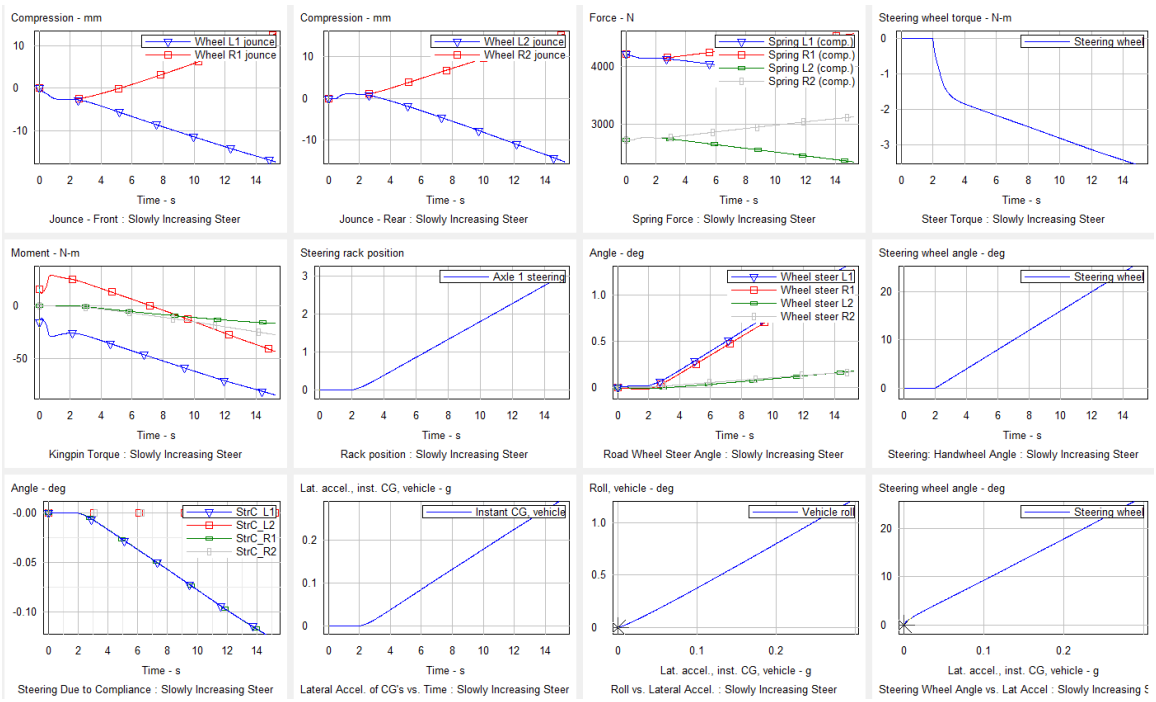

Figure 16. Plots for a slowly increasing steer test at 80 km/h.

This type of test can be used to confirm vehicle behavior at several levels of detail. The overall responsiveness is shown with lateral acceleration A_y, and cross-plots of other variables against lateral acceleration (steer vs. A_y, roll vs. A_y, etc.). It shows side-to-side load transfer (suspension jounce, spring forces, etc.). It shows components of the suspension and steering system that contribute to the steering of the road wheels, and some of the reaction torques (kingpin, steering wheel, etc.).

The slowly increasing steer test is also used in several regulations and is supported by the ISO validation standards mentioned earlier: ISO 19364 "*Passenger cars — Vehicle dynamic simulation and validation — Steady-state circular driving behavior*" (2016) and ISO 19365 "*Passenger cars — Validation of vehicle dynamics simulation — Sine with dwell stability control testing*" (2016). (However, the rate of steering wheel angle is 13.5°/s in those examples; a lower rate of 2°/s is used here to focus on near-equilibrium conditions.)