Mechanical Simulation

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

CarSim and TruckSim Suspensions

Suspension Types

SAE J670 defines a suspension as “a system that provides ride freedom and kinematic control of the motions of the wheels…”.

VehicleSim (VS) Math Models in CarSim and TruckSim support many different suspension designs, using data that can be obtained from real or simulated kinematics and compliance (K&C) tests. This approach means that details of a specific design (linkage geometry, bushing properties, etc.) are not needed, because the behavior of the suspension as it affects vehicle response is captured with more generic system-level parameters and nonlinear tables. This approach reduces the number of suspension models to just a few types, based on fundamental kinematical behavior.

Independent and Solid Axle Suspensions in CarSim and TruckSim

CarSim and TruckSim share two basic types of suspension models: *independent* and *solid axle*. Both suspension models include two “large” independent motions per suspension (multiple centimeters of vertical wheel travel).

The independent suspension is one in which the two independent motions are vertical displacements — commonly called *jounce* — of the two wheels. Along with the two large motions that serve to suspend the sprung mass, this suspension includes kinematical relationships between other motions (lateral displacement, longitudinal displacement, and steer, inclination, and pitch angles) for each wheel. The kinematical motions are handled with 2D Configurable Functions that define the secondary motions for one side as functions of wheel jounce from both sides. Because the kinematical effects can potentially involve jounce from both wheels, the independent suspension is sometimes called *generic/independent*.

The solid axle suspension has a rigid axle or a linkage system that causes both wheels to roll together. In this case, the two independent motions are axle jounce (vertical motion at the axle center) and axle roll. With solid axles, the kinematical relationships are characterized with 2D Configurable Functions that define the secondary motions for a solid axle (longitudinal and lateral movement, pitch and steer angles) as functions of axle jounce and axle roll.

In both cases (generic/independent and solid axle), lateral and longitudinal forces are transmitted to the sprung mass along lines of action perpendicular to the path of constrained motion, to determine load transfer due to suspension kinematical properties.

Both kinds of suspension models also include compliance effects, where the secondary motions are also influenced by tire forces and moments applied to both wheels of the suspension. The solid axle model includes additional compliance effects if it is steered vs. non-steered. In CarSim, the version of the solid axle with additional compliance is referred to as the *solid axle with wheel compliance* when used on the rear axle.

CarSim Virtual Steering Axis Suspension

CarSim also includes a *virtual steering axis* suspension (sometimes called *double ball joint*), which was introduced in version 2020.1.

The virtual steering axis suspension is an independent front suspension which has no fixed kingpin axis. These suspensions arise when a lower or upper control arm (wishbone) is split into two tension-compression links, resulting in double ball joints at the wheel-side of the control arm. In this case, the steering axis is no longer representable by fixed geometric parameters such as kingpin inclination or caster angle. Instead, the steering of the wheel relative to the sprung mass is about an instantaneous steering axis given by the linkage's geometry. In CarSim, this suspension type is modeled at the system level by requiring the suspension kinematics tables to be entered as 2D tables of jounce and steering rack travel. To avoid confusion with the generic/independent suspension, the virtual steering axis suspension is rarely called an independent suspension in CarSim; the virtual steering axis name is preferred.

CarSim Twist-Beam Suspensions

A *twist beam* suspension (also called *twist axle*) has a torsionally flexible structure linking the two sides. Lateral forces are transmitted to the sprung mass through lateral reactions at bushings attaching the structure to the chassis and through vertical reactions at the same bushings caused by the *twist* (structural deformation) of the beam. Because the twist beam relies on structural flexing, the motions of the two wheels are coupled more strongly than most independent suspensions, but not with the rigid connection of a solid axle.

CarSim supports three options for simulating twist beam suspensions:

1. The generic/independent suspension model can represent the twist beam with 2D kinematics and compliance tables. The way to measure and input the data for this twist beam model is described in a separate Technical Memo: *Twist Beam Suspensions: Using 2D Tables*. This capability was introduced in version 2018.0, and, since it is based on the generic/independent model, works in both CarSim and TruckSim.

2. A custom set of CarSim screens uses the generic/independent suspension model and adds VS Commands to set compliance coefficients. This option is named Twist Beam 2016 and was introduced in version 2016.0. It is described in a separate Technical Memo: *Twist Beam Suspensions: Using VS Commands* (memo not available in TruckSim).

3. A custom suspension model based on a solid axle was used in CarSim in versions prior to 2016.0 and is named Twist Beam (Legacy).

The modeling of twist beam suspension kinematics in CarSim is described in the **Twist Beam** sections (page 26).

Vehicle and Suspension Layout Codes

The type of suspension used for a simulation is indicated in the vehicle layout code that appears on the **Run Control** screen in the vehicle dataset link (Figure 1).

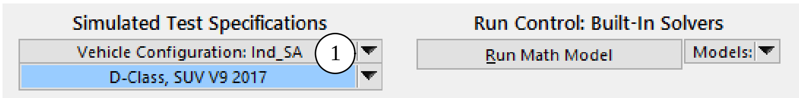


Figure 1. Vehicle layout code on the Run Control screen in CarSim.

In CarSim, `SA` indicates a solid-axle suspension, `Ind` is an independent suspension, `Vir` is a virtual steering axis suspension, and `Twist` is a legacy twist beam (the other twist beam options use the `Ind` code). For example, the code `Ind_SA` 1 indicates a front independent suspension and a solid-axle rear suspension; the code `Ind_Ind` indicates independent suspensions on the front and rear. In TruckSim, `S` indicates solid-axle, and `I` indicates independent.

Drop-down lists on the **Vehicle: Assembly** screen (CarSim), all vehicle lead unit screens (TruckSim), and all trailer screens (CarSim and TruckSim) are used to choose the suspension type for each axle (Figure 2). The selection of available types is different between CarSim and TruckSim, and according to the axle position on the vehicle.

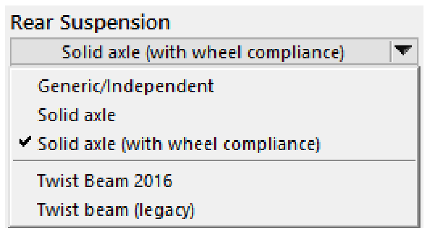


Figure 2. Selection of a suspension for a rear axle in CarSim.

A suspension is described by assembling datasets, each describing a subset of the suspension properties. The datasets are organized in over 20 libraries. Many of these involve components or properties that are used for all types of suspension: springs, dampers, bump stops, compliances affecting wheel orientation, etc.

All types of suspension are typically organized using two top-level datasets: one for kinematical data and one for compliance data. Links to the top-level data screens are shown on the vehicle screen based on the selection from the suspension drop-down list (Figure 2).

The VehicleSim Hysteretic Spring Model

As implied by the name, the main purpose of a suspension is to suspend the sprung mass, providing some isolation from the wheel motions due to road roughness.

Each suspension in CarSim and TruckSim includes six springs that produce force as the result of spring compression:

1. Two suspension springs (one per side).
2. Two jounce bump stops (one per side).
3. Two rebound bump stops (one per side).

The suspension spring models in CarSim and TruckSim use a hysteretic spring model based on concepts proposed in the 1980s at the University of Michigan Transportation Research Institute (UMTRI). Figure 3 shows spring force vs. deflection obtained during a maneuver in TruckSim. This behavior is typical of that observed in laboratory measurements of leaf springs, and of suspension systems in general.

The figure shows two limit curves of force vs. deflection: one when the load is increasing (during compression), and one when the load is decreasing (extension). Forces are always higher during compression due to inter-leaf friction resisting the motion. For example, in the figure, a vertical line is drawn for a deflection of 96.5 mm. During compression, the spring force is 52,600 N; during extension, the force is 42,400 N. The difference (10,200 N) is due to friction. The friction is half the difference in spring force (e.g., 5,100N); the compressive force limit is the spring without friction plus the friction; the extension force limit is the spring without friction minus the friction.

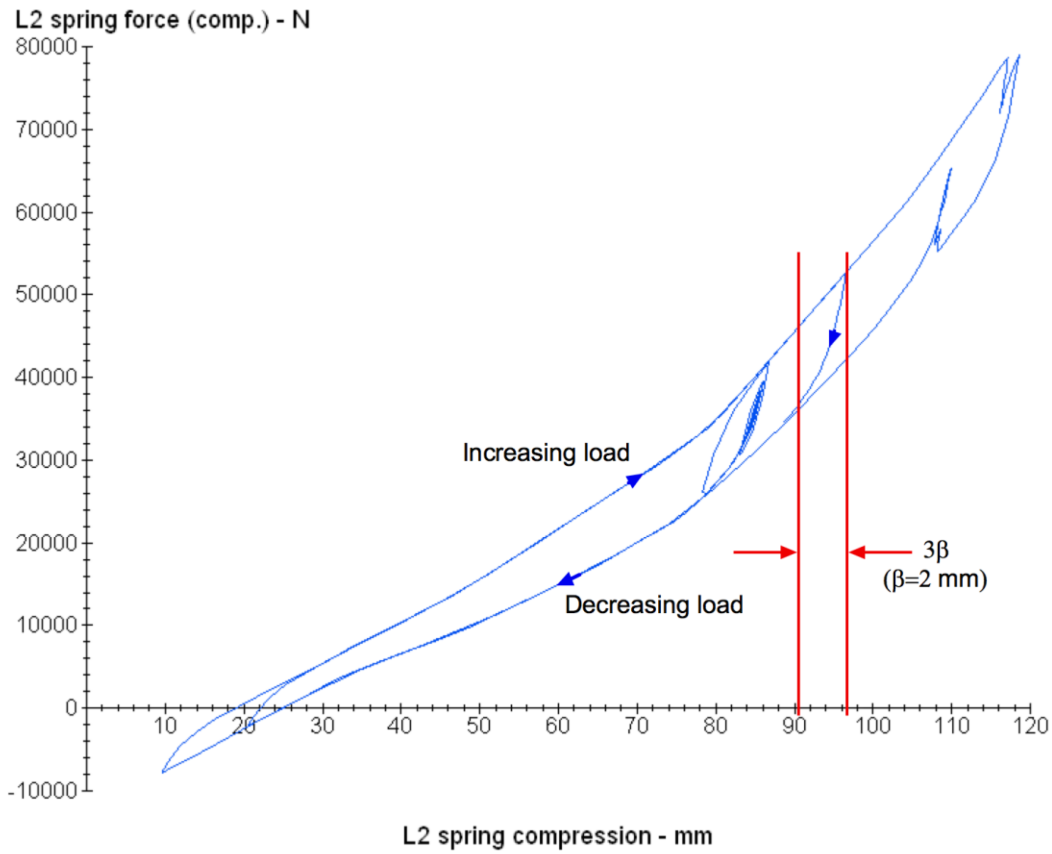


Figure 3. Simulation results showing spring force vs. deflection.

Friction is a significant factor in heavy truck suspensions and is present to a lesser extent in nearly all vehicle suspensions. It is important because it represents energy removed from the system that is not controlled by the dampers.

VS spring models are typically described with two force/deflection curves: one that applies during loading (compression), and another that applies during unloading (rebound).

When a reversal occurs, such as the one between the two vertical red lines in Figure 3, the force does not jump instantly from one limit to the other. A certain amount of deflection must occur for the force to approach the other limit. The math models use a spatial equivalent of a time constant called β to characterize the transition. The deflection needed to cover 95% of the force difference between the two limits is defined in the math model as 3β. For example, the TruckSim spring used to generate the example in Figure 3 has a model parameter of β = 2 mm, so the two vertical red lines spaced horizontally by 3β (6 mm) account for 95% of the force change.

The equations for this model require that the upper force curve (increasing load) never drop below the lower force curve (decreasing load) for any possible deflection.

Jounce and Component Compression

Jounce is a term used in automotive engineering to refer to vertical suspension deflection. It is the upward vertical displacement of the wheel center from some reference position, projected onto the Z axis of the sprung mass coordinate system. Downward displacement is called *rebound*, *droop*, or negative jounce.

The deflections of springs, dampers, jounce bumpers and rebound stops, etc., are direct functions of suspension deflection. Deflections of these components must be properly related to suspension deflection in order to ensure that the simulation correctly reflects the force-deflection behavior of the wheels. The reference point for zero jounce in VS math models may be determined in one of two ways, each with associated advantages and disadvantages.

The choice of which method is used to define zero jounce should be made based on the K&C measurement methods used, the specific studies to be performed, and the conventions for defining these terms in your company.

Sprung Mass Coordinate System

Most of the vehicle components in a CarSim or TruckSim vehicle model are located using a coordinate system fixed in the sprung mass for each vehicle unit. The sprung mass has X-Y-Z axes (X is forward, Z is up), with an origin that is typically located at the lateral center of the sprung mass, at the longitudinal location of the front suspension, and with a vertical location that is lower than the wheel centers in most loading conditions (Figure 4, Figure 5).

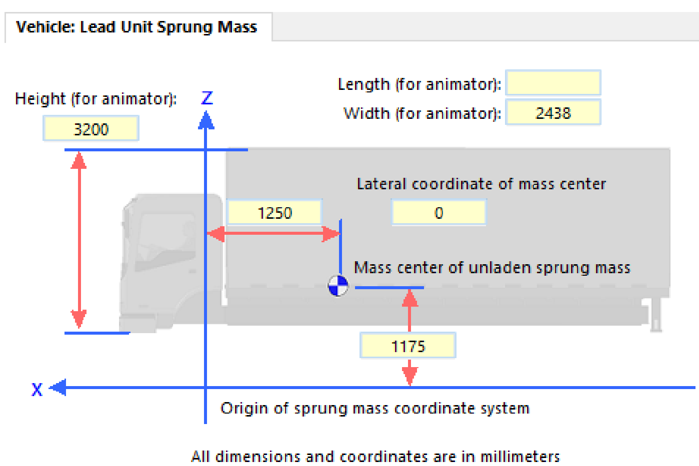


Figure 4. Part of the Sprung Mass screen in TruckSim.

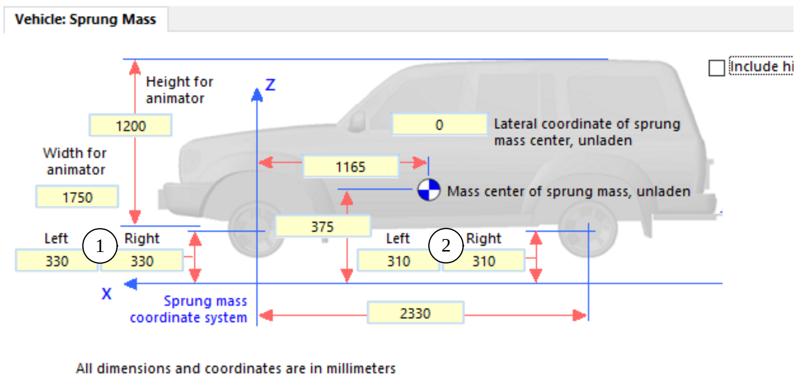


Figure 5. Wheel center height for each wheel is set from a sprung mass screen in CarSim.

Because the coordinate system is fixed in a rigid body of the multibody vehicle model, it remains valid regardless of how the body is oriented relative to the ground and global coordinate system.

Note CarSim and TruckSim do not have a fixed requirement for the origin of a sprung mass coordinate system. The only requirements are that X is forward, Z is up, and Y is to the left.

The vertical height of a point fixed in the sprung mass to the ground depends on the load conditions and the tire properties, so it is generally not practical to try to set the origin at the ground.

Although there is no built-in requirement for where the origin of the sprung mass coordinate system is located, it is essential that all coordinates for points in the sprung mass coordinate system use the same definition. The wheel center heights define suspension jounce in the design load, and therefore determine how physical measurements of spring deflections and suspension kinematical effects are linked to a CarSim or TruckSim math model.

Wheel Center Height

The first item to consider in defining the suspension motion and the meaning of jounce is the *wheel center height*, a reference Z coordinate specified for each wheel with a parameter `H_WC`. The wheel center height parameter `H_WC` can be set for each wheel in CarSim in data fields in the sprung mass screens (1 and 2, Figure 5). Figure 6 shows the right-click popup note for `H_WC` at the right-rear wheel.

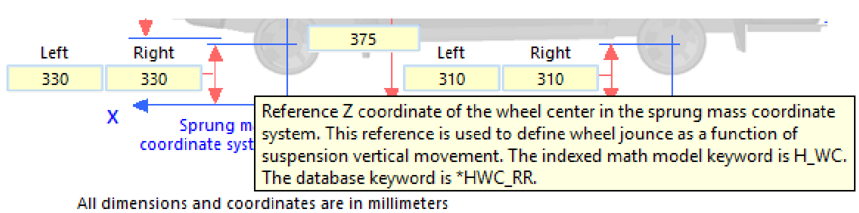


Figure 6. Right-click notes for a reference Z coordinate of a wheel.

Because sprung masses in TruckSim may be used with many options for axles, the wheel center heights in TruckSim are set on the suspension kinematic screens (Figure 7).

To provide backward compatibility for legacy datasets, some of the suspension screens in CarSim also have a data field to specify the wheel center height (usually hidden unless a checkbox is checked).

Note If the fields for `H_WC` on both screens are filled in CarSim, the values on the vehicle sprung mass screen take precedence and are used in the simulation.

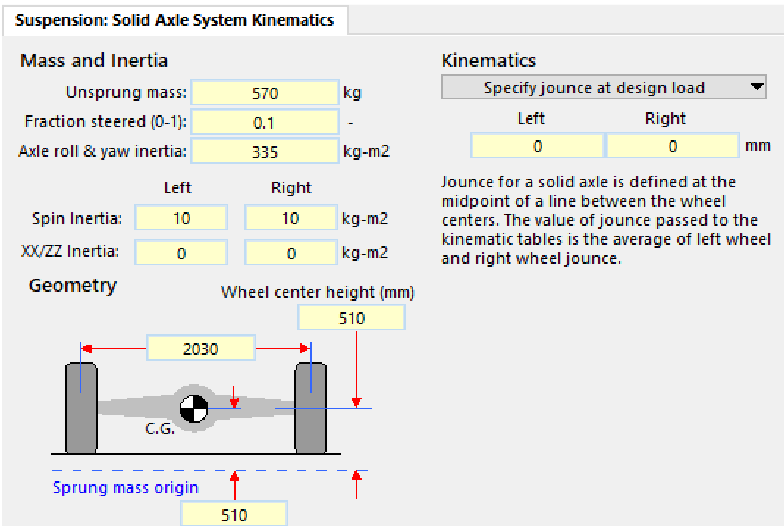


Figure 7. Wheel center height is specified on the kinematic suspension screens in TruckSim.

Design Load

Wheel center height is the vertical height in the sprung mass coordinate system Z direction of the wheel center relative to the sprung mass coordinate reference origin when the vehicle is loaded to its *Design Load*. Design Load in a VS Math Model means the weight of the sprung mass matches that specified on the vehicle sprung mass screen (no payload or hitch load), and the roll and pitch angles are zero. The load is distributed between axles according to their longitudinal locations and divided equally between the left and right sides.

Design Load from the vehicle sprung mass screen is usually not the same as *curb load*, a term usually used to refer to the vehicle loaded with some fuel (typically one-half tank) and some occupants (typically driver and passenger). To build a vehicle up to curb load (or any other load condition), payloads can be used for fuel and occupant loads, and any other loads such as cargo. For more information about design loads, please see the Technical Memo: *The Design Load Condition*.

Note Calculating the initial load on each wheel is a *statically indeterminate* problem further complicated by friction, potentially uneven ground, and potentially asymmetric features. This means the initial wheel loads cannot be calculated easily. VS Math Models include some simplifying assumptions used to estimate initial loads. They assume initial left/right symmetry of weight. They also assume an initial distribution of weight between tandem axles (as in TruckSim or in a CarSim multi-axle trailer) defined by a user parameter. In cases of very asymmetric vehicles, you may need to perform several iterative analyses to find settings that represent your configuration at initialization.

The ambiguity only applies to the initial condition. Once the simulation starts, the full dynamic equations provide a realistic response. Running a dynamic simulation on a flat surface at low speed is one way to find load conditions that account for all nonlinear and asymmetric effects.

For more information about the initial condition, please see the Technical Memo: *Initialization in CarSim and TruckSim*.

Jounce at Design Load

The second item to consider in defining suspension motion is the setting of a drop-down control on the various suspension kinematics screens (Figure 8). Use it to select the method for defining the value of jounce assigned to the position of a wheel at design load with the parameter `JNC_DESIGN`.

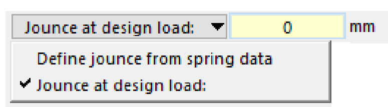


Figure 8. Drop-down list for choosing method of defining jounce at design load.

Jounce at design load is selected for most simulations. When chosen, the browser reveals one or two data fields for specifying jounce when the vehicle is at design load. (The screens for independent suspension kinematics have a checkbox to specify separate left and right properties; when not checked, a single field is shown, and the value is used to set the left- and right-side `JNC_DESIGN` parameters with the same value.) These numbers apply offsets for suspension kinematic tables to obtain camber, toe, etc. When set to zero, jounce means “the vertical displacement from design position”. When set to a nonzero value, the effect is to adjust the ride height.

For example, to simulate a vehicle modified to raise its ride height 20 mm by increasing spring length while keeping all suspension components unchanged, you need to push the wheels down and away from the chassis by 20 mm. To do this, enter -20 mm in the data field(s) for jounce at design load (causing the suspension settings at this position to reflect 20 mm of suspension rebound or droop at the design load position). The wheel center heights must also be reduced by 20 mm. The kinematic tables would be unchanged.

When *Define jounce from spring data* is selected, the design spring load is used to perform an inverse table-lookup into the spring force table, obtaining the spring compression at that load. The spring compression is divided by the mechanical ratio for that spring to obtain a number and assigns it to jounce at design load. That value is used to obtain the suspension settings at the design load position from the kinematic tables. It has no effect on the wheel center height at design load; it only sets the number assigned to the variable “jounce” at that position, and the camber, toe, etc. set for that position. This control setting is most often used together with a mechanical ratio of 1.0 (when the ratio is 1.0, the spring compression is always equal to the jounce at the wheel center).

Spring Compression

The next item to consider in setting up suspension deflection is the physical meaning of spring compression on the **Suspension: Spring** screen. If the spring data is obtained from a K&C test, it is typical to define zero compression as the design position of the vehicle. The ratio of spring compression to wheel jounce is ordinarily set to 1.0, and the option *Define jounce from spring data* is selected, because the wheel rate data is used directly. In this case, a change to the spring should ensure that the curve passes through zero deflection at the same load as the original, or the values of jounce sent to the kinematic tables will be changed.

When spring data are obtained from design information or from a bench test, the choice of a point to be called zero deflection is somewhat arbitrary. In this case, the “Specify jounce at design load” is most often used, and the ratio of spring compression to wheel jounce reflects the actual suspension linkage behavior.

> **Note** When a linear coefficient is used to specify a spring rate, zero compression is defined to occur at zero force. When separate compression and extension tables are used (springs with hysteresis) for spring force and deflection, the initial deflection is taken at a point midway between the two curves at the design spring load.

If spring compression is set to a nonlinear function of wheel jounce (see the subsection Libraries for Nonlinear Compliance, Springs, and Dampers on page 57 for details), the *Specify jounce at design load* option **must be used**. While it is possible to create variable motion ratios with the other option (Define jounce from spring data) and still obtain correct results, it is generally an inconvenient process to ensure that all initial settings meet your intent, as the very tight interconnection makes it necessary to change several things if any single setting is changed.

Jounce and Rebound Stops

The next item to consider is the meaning of the information in the tables on the **Suspension: Jounce and Rebound Stops** screen. Jounce and rebound stops are additional spring components, usually made of an elastomer, that limit the range of suspension travel. Conceptually, they are implemented as springs that have zero force over some deflection, and then produce force when deflected beyond that point. The range over which they produce no force represents the displacement of a point on the line of action of the component before contact occurs. It may be helpful to think of a “virtual spring” in place of the jounce bumper that always connects the suspension and chassis but produces no force for some of its deflection. Each can have a unique mechanical ratio of spring compression to suspension compression. The calculated suspension jounce is multiplied by the mechanical compression ratio for the component, and that value is input to the table as compression. If a jounce bumper is installed on a damper, for example, its mechanical ratio is set to the same value as the damper, its table specifies zero force up to the damper deflection at which the jounce stop is contacted (typically the length of the exposed rod at design height, sometimes called *engagement travel* or *engagement length*), and the force beyond that represents the load/deflection curve *for the jounce bumper*. The forces and deflections written to the output file represent the deflection and load in the bumper itself, not the effects measured at the wheel center, unless the mechanical compression ratio is set to one.

The standard in VS Math Models is to output forces and deflections at each component (spring, damper, jounce bumper, rebound stop), as they occur *in the component*. Understanding the meaning of jounce, component compression, and mechanical compression ratio is important to ensure your intent is met. When using K&C data, it is most convenient to use measured wheel rate data and set all mechanical ratios to one, but care must be taken if components are changed that the new data also reflect performance of the component as measured at the wheel, taking all mechanical linkage effects into account. The compression of a suspension force component is always jounce multiplied by the mechanical compression ratio (or the output of a table for component deflection as a function of jounce in the case of nonlinear relationships). The initial value of jounce that is used may be specified directly at design load, or it may be obtained by reading the spring compression from the spring table at design load and dividing by the mechanical compression ratio.

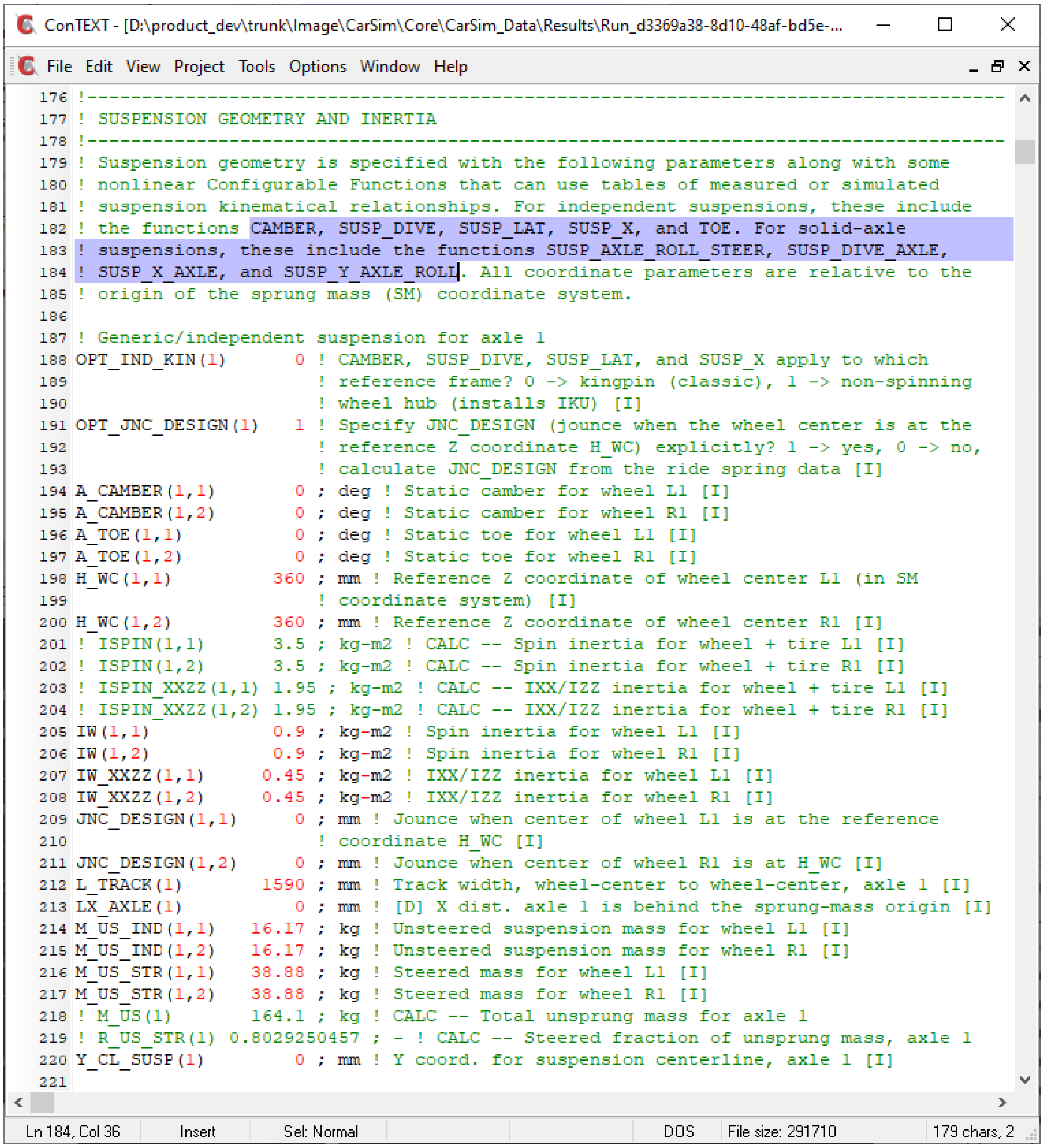
Screens for Suspension Geometry and Inertia

The GUI screens in CarSim and TruckSim are organized into two parts: the role of a suspension to "provide ride freedom" by suspending the sprung mass, and "to provide kinematic control of the motions of the wheels". This section covers information about the kinematics and the inertia properties of the unsprung masses associated with the suspension.

The parameters and summary information for the suspension kinematics are listed in a section of the Echo file, along with inertia properties (Figure 9). In this example, axle 1 is a generic/independent suspension. As in other sections of the Echo file, the root names for relevant Configurable Functions are listed at the top of the section (line 182-184).

The kinematical properties of a suspension are assembled on a screen such as the **Suspension: Independent System Kinematics** screen (Figure 10). This screen also contains a few mass and inertia properties.

The kinematics of the suspension linkages are described by the lateral and longitudinal motions of the wheel or axle as the suspension deflects vertically. The lateral movement primarily affects the transfer of tire lateral force to the body and the resulting body roll. The longitudinal movement primarily affects the transfer of tire longitudinal force to the body and the resulting body pitch. These effects are sometimes described using concepts such as roll centers and "side view swing arms," particularly when performing simple analyses. The VS Math Models do not use fixed roll center points or swing arm approximations—the full 3D position and orientation of the wheel or axle relative to the sprung mass is specified instead, using the suspension kinematics input tables. Instantaneous characteristics like front and side view swing arms are implied by the partial derivatives of these suspension kinematics input tables.



```
ConTEXT - [D:\product_dev\trunk\Image\CarSim\Core\CarSim_Data\Results\Run_d3369a38-8d10-48af-bd5e-...
File Edit View Project Tools Options Window Help
!------------------------------------------------------------------------------------
! SUSPENSION GEOMETRY AND INERTIA
!------------------------------------------------------------------------------------
! Suspension geometry is specified with the following parameters along with some
! nonlinear Configurable Functions that can use tables of measured or simulated
! suspension kinematical relationships. For independent suspensions, these include
! the functions CAMBER, SUSP_DIVE, SUSP_LAT, SUSP_X, and TOE. For solid-axle
! suspensions, these include the functions SUSP_AXLE_ROLL_STEER, SUSP_DIVE_AXLE,
! SUSP_X_AXLE, and SUSP_Y_AXLE_ROLL. All coordinate parameters are relative to the
! origin of the sprung mass (SM) coordinate system.

! Generic/independent suspension for axle 1
OPT_IND_KIN(1)        0 ! CAMBER, SUSP_DIVE, SUSP_LAT, and SUSP_X apply to which
                        ! reference frame? 0 -> kingpin (classic), 1 -> non-spinning
                        ! wheel hub (installs IKU) [I]
OPT_JNC_DESIGN(1)     1 ! Specify JNC_DESIGN (jounce when the wheel center is at the
                        ! reference Z coordinate H_WC) explicitly? 1 -> yes, 0 -> no,
                        ! calculate JNC_DESIGN from the ride spring data [I]
A_CAMBER(1,1)         0 ; deg ! Static camber for wheel L1 [I]
A_CAMBER(1,2)         0 ; deg ! Static camber for wheel R1 [I]
A_TOE(1,1)            0 ; deg ! Static toe for wheel L1 [I]
A_TOE(1,2)            0 ; deg ! Static toe for wheel R1 [I]
H_WC(1,1)           360 ; mm ! Reference Z coordinate of wheel center L1 (in SM
                        ! coordinate system) [I]
H_WC(1,2)           360 ; mm ! Reference Z coordinate of wheel center R1 [I]
! ISPIN(1,1)        3.5 ; kg-m2 ! CALC -- Spin inertia for wheel + tire L1 [I]
! ISPIN(1,2)        3.5 ; kg-m2 ! CALC -- Spin inertia for wheel + tire R1 [I]
! ISPIN_XXZZ(1,1) 1.95 ; kg-m2 ! CALC -- IXX/IZZ inertia for wheel + tire L1 [I]
! ISPIN_XXZZ(1,2) 1.95 ; kg-m2 ! CALC -- IXX/IZZ inertia for wheel + tire R1 [I]
IW(1,1)             0.9 ; kg-m2 ! Spin inertia for wheel L1 [I]
IW(1,2)             0.9 ; kg-m2 ! Spin inertia for wheel R1 [I]
IW_XXZZ(1,1)       0.45 ; kg-m2 ! IXX/IZZ inertia for wheel L1 [I]
IW_XXZZ(1,2)       0.45 ; kg-m2 ! IXX/IZZ inertia for wheel R1 [I]
JNC_DESIGN(1,1)       0 ; mm ! Jounce when center of wheel L1 is at the reference
                        ! coordinate H_WC [I]
JNC_DESIGN(1,2)       0 ; mm ! Jounce when center of wheel R1 is at H_WC [I]
L_TRACK(1)         1590 ; mm ! Track width, wheel-center to wheel-center, axle 1 [I]
LX_AXLE(1)            0 ; mm ! [D] X dist. axle 1 is behind the sprung-mass origin [I]
M_US_IND(1,1)     16.17 ; kg ! Unsteered suspension mass for wheel L1 [I]
M_US_IND(1,2)     16.17 ; kg ! Unsteered suspension mass for wheel R1 [I]
M_US_STR(1,1)     38.88 ; kg ! Steered mass for wheel L1 [I]
M_US_STR(1,2)     38.88 ; kg ! Steered mass for wheel R1 [I]
! M_US(1)         164.1 ; kg ! CALC -- Total unsprung mass for axle 1
! R_US_STR(1) 0.8029250457 ; - ! CALC -- Steered fraction of unsprung mass, axle 1
Y_CL_SUSP(1)          0 ; mm ! Y coord. for suspension centerline, axle 1 [I]

Ln 184, Col 36 | Insert | Sel: Normal | DOS | File size: 291710 | 179 chars, 2
```

Figure 9. Suspension Geometry and Inertia parameters listed in the Echo file.

User Settings for Generic/Independent and Solid Axle Suspensions

The controls in Figure 10 are numbered such that (1) - (11) exist on the Kinematics screens for both generic/independent and solid axles in CarSim and TruckSim. They are described in this section.

All of the parameters and Configurable Functions that are set on a suspension kinematics screen are indexed to the axle number as indicated by the current value of the system parameter `IAXLE`. For models with trailers, they are also indexed to the unit as indicated by the current value of the system parameter `IUNIT`. Most of the properties can be applied to either the left or right side. The context is specified in the parsfile with the system parameter `ISIDE` (1 = left, 2 = right).

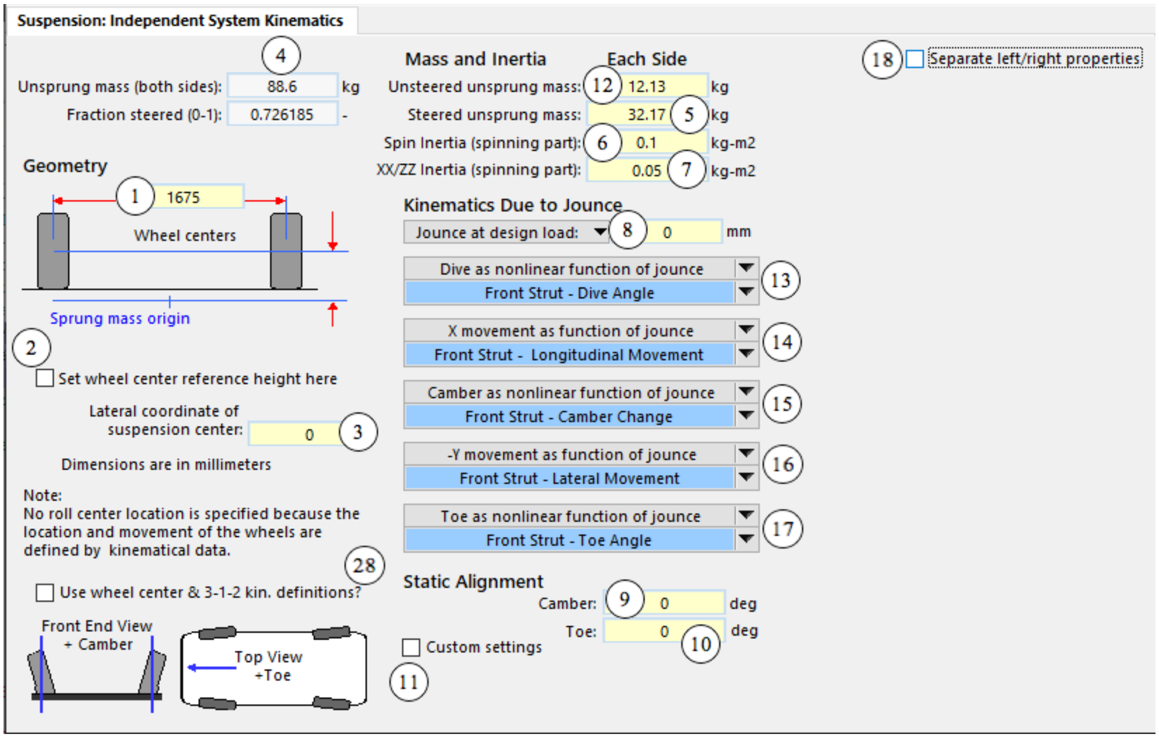


Figure 10. The CarSim Suspension: Independent System Kinematics screen.

Geometry

(1) Track width (keyword = `L_TRACK`) when the wheel centers are at the reference heights.

(2) Reference Z coordinate of the wheel spin axes in the sprung mass coordinate system (keyword = `H_WC`). This defines the relationship between the location of the wheel relative to the sprung mass and the jounce used to define toe and camber effects.

In CarSim, the data field is not shown unless the adjacent box is checked. The preferred location for setting wheel center height is on the **Vehicle: Sprung Mass** screen where other dimensions and coordinates in the sprung mass coordinate system are specified (Figure 5, page 7). The data field exists on this screen to provide backward compatibility with data from older versions of CarSim. If the wheel heights are specified both here and on the **Vehicle: Sprung Mass** screen, the data from the **Vehicle: Sprung Mass** screen takes priority.

In TruckSim, the field for the Reference Z coordinate is always visible (Figure 7, page 8).

Considerations in setting wheel center height were discussed earlier (page 7).

(3) Lateral coordinate of the suspension center (keyword = `Y_CL_SUSP`). Normally zero, this can be given a non-zero value if the wheels are not located symmetrically about the longitudinal centerline of the vehicle sprung mass.

Mass and Inertia

(4) Unsprung mass (keyword = `M_US`). This mass includes the wheels, tires, brakes, and all parts that move vertically with the wheel as the suspension deflects. This is a legacy parameter used

in versions prior to 2019.1; it is now calculated by summing masses from this screen and also the Tire screen. It is listed in the Echo file as read-only (line 218, Figure 9) and cannot be set directly.

Fraction Steered (keyword = R_US_STR) is the portion of the unsprung mass that rotates about the kingpin axis when the wheels are steered. This is also legacy parameter used in versions prior to 2019.1 and is now calculated by summing steered masses from this screen and the **Tire** screen and dividing by M_US. It is listed in the Echo file as read-only (line 219, Figure 9) and cannot be set directly.

> **Note** The values of M_US and R_US_STR shown in the Echo file include the masses from the linked **Tire** datasets and may differ from the values shown on this screen that do not account for tire mass.

(5) Steered unsprung mass on one side (keyword = M_US_STR). This mass includes the parts of the suspension that are steered, not including the mass of the tire and rim that is specified on the **Tire** screen.

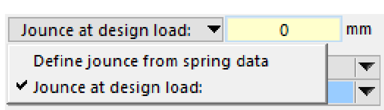
(6) Spin moment of inertia for non-removable parts that spin (hub, brake rotor, etc.) (keyword = IW). The total spin inertia for each wheel is the sum of this value plus the inertia due to the tire/rim as specified on the **Tire** screen. The total spin inertias for each side (ISPIN) are shown in the Echo file (lines 201 and 202).

(7) Yaw and roll moment of inertia for non-removable parts that spin (keyword = IW_XXZZ). The same value is applied to both the yaw (Izz) and roll (Ixx) moments of inertia for the wheel. The total yaw/roll moment inertia for each wheel is the sum of this value plus the inertia due to the tire/rim as specified on the **Tire** screen. The total inertias for each side (ISPIN_XXZZ) are shown in the Echo file (lines 203 and 204).

Kinematics Due to Jounce

The mechanics of a suspension cause movement of the wheels when jounce changes. These movements are commonly measured during K&C testing. The suspension models have several Configurable Functions that calculate motion variables from jounce. This requires that all references to jounce use the same definition of jounce.

(8) Pull-down control for choosing between two jounce definitions:



When the first option is chosen, the jounce at the design load condition is defined as the jounce in the spring table that corresponds to the spring force needed to support the sprung mass in the design load condition.

When the second option is chosen, an adjacent data field is displayed in which you specify the jounce when the vehicle is in the design load condition (keyword = OPT_JNC_DESIGN).

The first option links the jounce definition to the spring datasets. It is convenient if you do not change those datasets much. In this case, you can change the sprung mass properties without worrying about the nonlinear effects of jounce.

The second option links the jounce definition to the wheel-center design height (`H_WC`). It is convenient if you do not change the design height much. In this case, you can change spring properties without making corresponding adjustments to the toe and camber tables.

Recall that an earlier section discussed some of the considerations in choosing between these two options (page 9).

Kingpin Geometry and Spin Axis

For systems with fixed kingpin geometry, the steering axis is defined by geometric parameters that are set on the steering system screen using an X-Y-Z coordinate system (reference frame) attached to the knuckle (X forward, Z up) with origin at the wheel center. The orientation of the wheel spin axis in the same knuckle coordinate system is specified on the suspension kinematics screen, using static toe and camber angles:

(9) Static camber for each wheel (keyword = `A_CAMBER`). This angle together with the static toe (10) describes the fixed relationship between the orientation of the wheel spindle (spin axis) and the kingpin (steer axis).

(10) Static toe for each wheel (keyword = `A_TOE`). This angle together with the static camber (9) describes the fixed relationship between the orientation of the wheel spindle (spin axis) and the kingpin (steer axis).

This completes the relationship between the kingpin axis and spin axis. This relationship is fixed in the coordinate system of the knuckle, regardless of the position or orientation of the knuckle with respect to the sprung mass.

> **Note** Non-steered systems use a vertical axis through the wheel center to model the rotation corresponding to toe change. Setting static camber and static toe using (9) and (10) will change the relationship between the spin axis and this vertical axis, whereas adding an offset to the camber or toe kinematics configurable functions will not. Either approach is viable.

Custom Settings

(11) Options for custom settings. Use the checkbox to display two miscellaneous links (independent and twist beam) or one link and a miscellaneous yellow field (solid axle). These controls have no predefined purpose, but they can be used to specify parameters for extensions to the model or information for the animator. They are not displayed unless the box is checked.

Suspension: Independent System Kinematics

Following are the kinematic settings that are unique for the independent suspension screen. These numbers also refer to Figure 10, page 13.

12 Unsteered unsprung mass on one side (keyword = `M_US_IND`). This mass includes the parts of the suspension that move vertically with the suspension, but which are not steered. Even if the suspension is not steered, this should not include the mass of the removable part specified on the Tire screen.

13 Setting to specify the rotation of the unsprung mass about the sprung mass Y-axis (pitch) as a Configurable Function of suspension jounce. *Dive angle* is positive when the unsprung mass makes a positive right-hand rotation about the Y axis (rotates forward at the top).

A drop-down control specifies whether the function should use a linear coefficient (keyword = `SUSP_DIVE_COEFFICIENT`) specified in a yellow field, or a blue link to a **Suspension: Dive Angle (Caster Change)** dataset, as shown in the figure. The Configurable Function defines dive of the unsprung mass as a function of jounce on the side of the wheel, and possibly also the jounce of the wheel on the other side.

How dive is applied in the generic/independent model depends on the checkbox 28.

- When 28 is unchecked, dive gives a side view pitch change which is mathematically equivalent to caster change (up to sign) if the kingpin geometry is vertical (caster = kingpin inclination = zero). Otherwise, it is approximately equal in magnitude to caster change.
- When 28 is checked, dive gives the wheel spin change, assuming locked, outboard brakes. This is the effect from the suspension's jounce kinematics. For steered systems, a nonzero steering gear output will produce additional wheel spin change for a typical kingpin geometry.

14 Setting to specify the longitudinal movement of the unsprung mass as a Configurable Function of suspension jounce. Longitudinal movement is positive when the unsprung mass moves forward (sprung mass +X direction).

A drop-down control specifies whether the function should use a linear coefficient (keyword = `SUSP_X_COEFFICIENT`), or a blue link to a **Suspension: Longitudinal Movement** dataset, as shown in the figure. The Configurable Function defines longitudinal movement for an unsprung mass as a function of jounce on the side of the unsprung mass, and possibly also the jounce of the wheel on the other side.

How longitudinal movement is applied in the generic/independent model depends on the checkbox 28.

- When 28 is unchecked, longitudinal movement is applied to the reference frame containing the kingpin axis, so the true wheel center longitudinal displacement from the suspension will differ slightly if the wheel has been rotated about the kingpin axis (such as the rotation due to toe vs. jounce data 17).
- When 28 is checked, longitudinal movement is applied at the wheel center. This is the effect from the suspension's jounce kinematics. For steered systems, a nonzero steering gear output will produce additional longitudinal movement for a typical kingpin geometry.

(15) Setting to specify camber as a Configurable Function of suspension jounce. Camber angle follows the usual sign convention (negative if the unsprung mass leans in at the top).

A drop-down control specifies whether the function should use a linear coefficient (keyword = `CAMBER_COEFFICIENT`), or a blue link to a **Suspension: Camber Angle** dataset, as shown in the figure. The Configurable Function defines change in camber for an unsprung mass as a function of jounce on the side of the unsprung mass, and possibly also the jounce of the wheel on the other side.

How camber is applied in the generic/independent model depends on the checkbox (28).

- When (28) is unchecked, camber gives a relative roll angle of the kingpin axis equivalent (up to sign) to a change in the kingpin inclination angle.
- When (28) is checked, camber applies to the angle between the wheel plane and the sprung mass Z axis. This is the effect from the suspension's jounce kinematics. For steered systems, a nonzero steering gear output will produce additional camber for a typical kingpin geometry.

(16) Setting to specify lateral movement of the unsprung mass as a Configurable Function of suspension jounce. Positive lateral motion occurs when the unsprung mass moves inward (for the right side this is the +Y direction; for the left side, the –Y direction).

A drop-down control specifies whether the function should use a linear coefficient (keyword = `SUSP_LAT_COEFFICIENT`), or a blue link to a **Suspension: Lateral Position** dataset describing how the lateral movement of the unsprung mass varies as a function of jounce on the side of the unsprung mass, and possibly also the jounce of the wheel on the other side.

How lateral movement is applied in the generic/independent model depends on the checkbox (28).

- When (28) is unchecked, this applies to the reference frame containing the kingpin axis, so the true wheel center lateral displacement due to the suspension will differ slightly if the wheel has been rotated about the kingpin axis (such as the rotation due to toe vs. jounce data (17)).
- When (28) is checked, lateral movement is applied at the wheel center. This is the effect from the suspension's jounce kinematics. For steered systems, a nonzero steering gear output will produce additional lateral movement for a typical kingpin geometry.

(17) Setting to specify wheel toe as a Configurable Function of suspension jounce. Positive toe means toe-in (same sign as steer for the right side; for the left side, opposite sign of steer).

A drop-down control specifies whether the function should use a linear coefficient (keyword = `TOE_COEFFICIENT`), or a blue link to a **Suspension: Toe Angle** dataset, as shown in the figure. The Configurable Function defines change in toe for a wheel as a function of jounce on the side of the wheel, and possibly also the jounce of the wheel on the other side.

How toe is applied in the generic/independent model depends on the definition of steer set on the steering screen (in vehicle coordinates vs. about the kingpin axis, `OPT_STEER_DEF`).

1. If OPT_STEER_DEF = 1 (default), the toe/steer is about the vehicle (sprung mass) Z axis, matching the measurement convention often used in K&C tests.
2. If OPT_STEER_DEF = 0, the toe/steer is about the kingpin axis.

The model uses steer/toe about the kingpin axis internally, but most test data is made from the vehicle Z axis. The first option always converts data from TOE and related tables in the steering system from the vehicle Z to the kingpin axis. The second option has the model use the table data "as is."

The relationship defined here between jounce and toe is what would exist during K&C testing when the steering gear input (or output) is held at zero, or there is no steering gear. The wheel steers slightly due to the linkage kinematics, but there is no contribution from the driver. For steered systems, a nonzero steering gear output will produce additional steer according to the steer kinematics vs. steering gear output Configurable Function entered on the steering screen.

(18) Checkbox to specify whether different properties should be specified for the left and right sides of the suspension. When unchecked (as in the figure), the specified settings described above ((5) - (17) excluding (11)) are written twice in the dataset for use on both sides. When checked, separate settings are shown in the screen for the left and right sides.

(28) Option to use alternate definitions of the suspension kinematics input tables (keyword = OPT_IND_KIN). The dive, X movement, camber, and lateral movement data, entered with (13) through (16), can apply to the kingpin axis (box unchecked) or to the steered, non-spinning wheel hub (box checked). The alternate definition applies the rotations to the wheel hub in 3-1-2 order (steer-inclination-dive, where toe and camber give steer and inclination) and the translations to the wheel center point. The competing definitions were covered in this section, with additional coverage in the later section Suspension Kinematics in the Math Models, p. 30. Using this option requires that steer angles be defined in vehicle coordinates, which is an option selected on the steering screen.

Suspension: Solid Axle System Kinematics

With a solid axle suspension, the axle is modeled as a rigid body that has two degrees of freedom relative to the sprung mass of the vehicle unit. One is axle jounce and the other is axle roll. The axle movements are constrained to follow paths defined using this screen, by linking to datasets for 2D tables to define longitudinal movement, lateral movement, dive (pitch) angle, and axle steer angle as functions of axle jounce and axle roll. For most solid axle example datasets, these tables are used in 1D mode, where axle jounce is considered the primary independent variable for longitudinal movement and dive angle, and axle roll is considered the primary independent variable for lateral movement and steer angle. The specific axle movement definitions are covered in this section, with additional coverage in the later section Suspension Kinematics in the Math Models, p. 30.

In vehicle dynamics terminology, vehicle roll angle is the angle of the sprung mass about its X axis; thus, positive roll implies the sprung mass is leaning to the right. CarSim and TruckSim use a relative roll angle with the same sign convention, called *roll relative to axle*. In a K&C test where the sprung mass is fixed and vertical actuators move the wheels up and down, positive roll relative to axle implies that the right wheel goes up and the left wheel goes down.

Figure 11 shows a section of the Echo file with the parameters and summary information for the suspension kinematics and inertia properties for a solid-axle suspension. In comparing this section with the example shown earlier for an independent suspension (Figure 9, page 12), we see that they are similar. The generic/independent suspension section shows parameters `M_US_IND` for each side that are not part of the solid-axle suspension, and the solid-axle section shows parameters like `IA` and `M_US_AXLE` that are not part of the independent suspension.

```
! Solid-axle suspension for axle 2
OPT_JNC_DESIGN(2)    1 ! Specify JNC_DESIGN explicitly for axle 2? [I]
OPT_SUSP_Y_AXLE_ROLL(2) 0 ! SUSP_Y_AXLE_ROLL gives kinematic lateral displacement of
                          ! 0 -> axle X-Y reference frame (legacy), 1 -> axle
                          ! center point [I]
A_CAMBER(2,1)          0 ; deg ! Static camber for wheel L2 [I]
A_CAMBER(2,2)          0 ; deg ! Static camber for wheel R2 [I]
A_TOE(2,1)             0 ; deg ! Static toe for wheel L2 [I]
A_TOE(2,2)             0 ; deg ! Static toe for wheel R2 [I]
H_CG_AXLE(2)         370 ; mm ! Z coordinate of axle 2 CG [I]
H_WC(2,1)            370 ; mm ! Reference Z coordinate of wheel center L2 [I]
H_WC(2,2)            370 ; mm ! Reference Z coordinate of wheel center R2 [I]
IA(2)                 26 ; kg-m2 ! Yaw and roll moment of inertia, axle 2 [I]
IA_YY(2)               0 ; kg-m2 ! [D] Pitch moment of inertia, axle 2 [I]
! ISPIN(2,1)        3.61 ; kg-m2 ! CALC -- Spin inertia for wheel + tire L2 [I]
! ISPIN(2,2)        3.61 ; kg-m2 ! CALC -- Spin inertia for wheel + tire R2 [I]
! ISPIN_XXZZ(2,1) 2.01 ; kg-m2 ! CALC -- IXX/IZZ inertia for wheel + tire L2 [I]
! ISPIN_XXZZ(2,2) 2.01 ; kg-m2 ! CALC -- IXX/IZZ inertia for wheel + tire R2 [I]
IW(2,1)             1.01 ; kg-m2 ! Spin inertia for wheel L2 [I]
IW(2,2)             1.01 ; kg-m2 ! Spin inertia for wheel R2 [I]
IW_XXZZ(2,1)        0.51 ; kg-m2 ! IXX/IZZ inertia for wheel L2 [I]
IW_XXZZ(2,2)        0.51 ; kg-m2 ! IXX/IZZ inertia for wheel R2 [I]
JNC_DESIGN(2,1)        0 ; mm ! Jounce when center of wheel L2 is at H_WC [I]
JNC_DESIGN(2,2)        0 ; mm ! Jounce when center of wheel R2 is at H_WC [I]
L_TRACK(2)          1590 ; mm ! Track width, wheel-center to wheel-center, axle 2 [I]
LX_AXLE(2)          3261 ; mm ! X dist. axle 2 is behind the sprung-mass origin [I]
M_US_AXLE(2)         124 ; kg ! Unsteered mass for axle 2 [I]
M_US_STR(2,1)          0 ; kg ! Steered mass for wheel L2 [I]
M_US_STR(2,2)          0 ; kg ! Steered mass for wheel R2 [I]
! M_US(2)            178 ; kg ! CALC -- Total unsprung mass for axle 2
! R_US_STR(2) 0.3033707865 ; - ! CALC -- Steered fraction of unsprung mass, axle 2
X_CG_AXLE(2)           0 ; mm ! [D] X offset of M_US_AXLE CG from axle center [I]
! X_CG_SUSP(2)         0 ; mm ! CALC -- X coord. for total unsprung mass, axle 2 [I]
Y_CG_AXLE(2)           0 ; mm ! [D] Y offset of M_US_AXLE CG from track center [I]
! Y_CG_SUSP(2)         0 ; mm ! CALC -- Y coord. for total unsprung mass, axle 2 [I]
Y_CL_SUSP(2)           0 ; mm ! Y coord. for suspension centerline, axle 2 [I]
```

Figure 11. Portion of Echo file for solid-axle suspension.

The kinematic properties of a solid axle suspension are assembled on the **Suspension: Solid Axle System Kinematics** screen (Figure 12). As noted earlier, the controls numbered 1 - 11 also exist on the Kinematics screen for both independent suspensions and were described earlier.

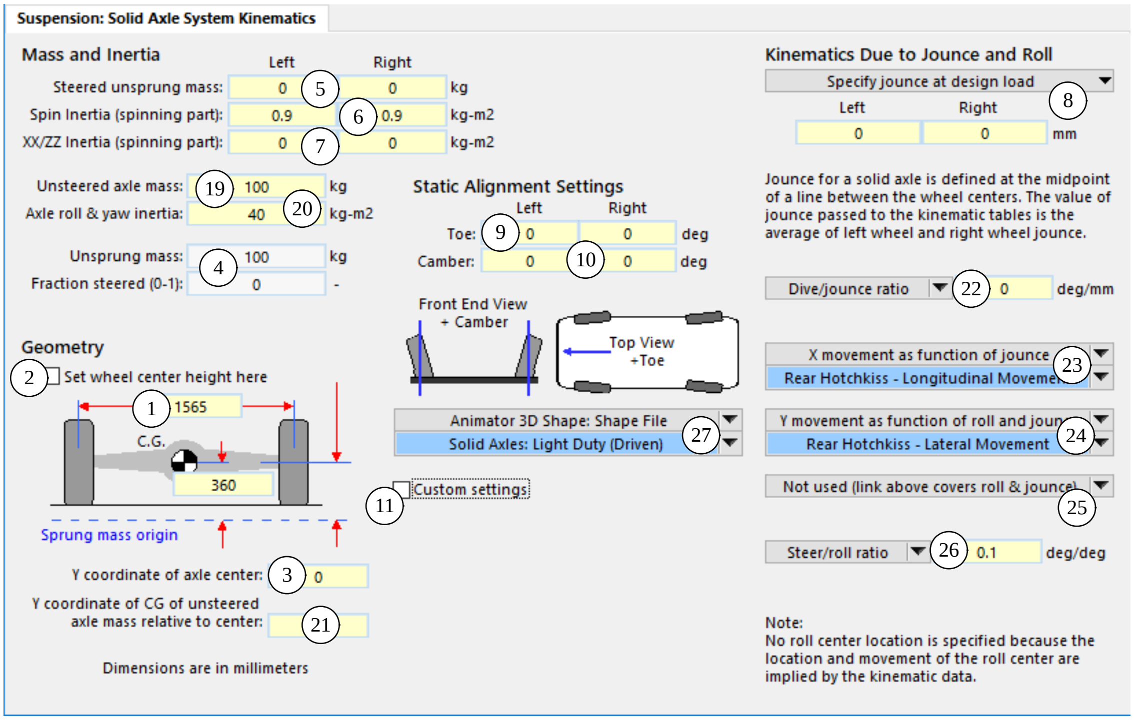


Figure 12. The Suspension: Solid Axle System Kinematics screen.

Following are the kinematic settings that are unique for the solid-axle suspension screen.

(19) Unsteered axle mass (keyword = `M_US_AXLE`). This is the solid body that connects the wheels.

(20) Moment of inertia of the solid body (keyword = `IA`). This moment is used for both roll and yaw.

(21) Y coordinate of the center of gravity for the solid body of the axle (`Y_CG_AXLE`). This is relative to the lateral center of the suspension, which itself located with Y coordinate `Y_CL_SUSP` (3) in the sprung mass coordinate system. The Y coordinate of the total unsprung mass is also calculated and listed in the Echo file (keyword = `Y_CG_SUSP`, line 251, Figure 11).

(22) Setting to specify the rotation of the unsprung axle mass about the sprung mass Y-axis (pitch) as a Configurable Function of axle jounce and possibly roll relative to axle. Dive angle is positive when the axle makes a positive righthand rotation about the Y axis (the axle rotates forward at the top).

A drop-down control specifies whether the function should use a linear coefficient (keyword = `SUSP_DIVE_AXLE_COEFFICIENT`) as shown in the figure, or a blue link to a **Suspension: Dive Angle (Solid Axle)** dataset which supports 1D functions of axle jounce, 2D functions of axle jounce and axle roll, or a combination of 1D functions (e.g., summing a 1D function of axle jounce with a 1D function of axle roll). For older datasets, there is also the option to link to a **Suspension: Dive Angle (Caster Change)** dataset, which, as of version 2021.1, is no longer recommended for solid axle suspensions.

Dive is applied to the solid axle body as a relative pitch angle with a sign convention opposite of caster change.

23 Setting to specify the longitudinal translation of the axle as a Configurable Function of axle jounce and possibly roll relative to axle. Longitudinal movement is positive when the axle center moves forward (sprung mass +X direction).

A drop-down control specifies whether the function should use a linear coefficient (keyword = `SUSP_X_AXLE_COEFFICIENT`), or a blue link to a **Suspension: Longitudinal Movement (Solid Axle)** dataset which supports 1D functions of axle jounce, 2D functions of axle jounce and axle roll, or a combination of 1D functions (e.g., summing a 1D function of axle jounce with a 1D function of axle roll). For older datasets, there is also the option to link to a **Suspension: Longitudinal Movement** dataset, as shown in the figure. The non-solid-axle dataset is no longer recommended for solid axle suspensions.

Longitudinal movement is applied to the center of the solid axle body.

24 Setting to specify axle lateral movement as a Configurable Function of roll relative to axle and possibly axle jounce. Positive lateral motion occurs when the axle moves to the left (+Y direction).

A drop-down control specifies whether the function should use a linear coefficient (keyword = `SUSP_Y_AXLE_ROLL_COEFFICIENT`), or a blue link to a **Suspension: Lateral Movement Due to Roll and Jounce** dataset which supports 1D functions of axle roll, 2D functions of axle roll and axle jounce, or a combination of 1D functions (e.g., summing a 1D function of axle roll with a 1D function of axle jounce).

The exact definition used in the VS Math Model for the solid axle lateral movement depends on the setting of the checkbox on the **Suspension: Lateral Movement Due to Roll and Jounce** dataset, Figure 13. For new datasets, we recommend checking the box to have the lateral movement table completely specify the axle center point's lateral displacement. When unchecked, the lateral movement table specifies part, but not all, of the axle center point's lateral displacement; there is a contribution from axle jounce in the direction established by axle roll. The unchecked setting is for backwards compatibility with legacy datasets.

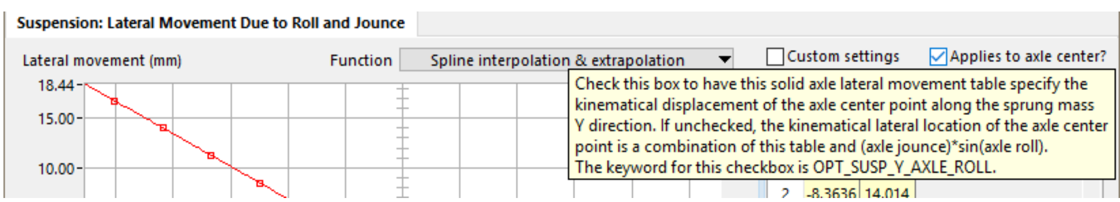


Figure 13. Choosing the solid axle lateral movement definition on the solid axle lateral movement due to roll and jounce library screen.

25 Setting to specify axle lateral movement as a Configurable Function of suspension jounce.

> **Note** Versions of CarSim and TruckSim prior to 8.2 did not support the option for using a 2D table with combined effects of roll and jounce. In those

older versions, this link was the only option for providing sensitivity to jounce. It is still provided for backward compatibility with legacy datasets.

When making new datasets, we recommend putting all lateral movement data for a solid-axle suspension in a single dataset from the **Suspension: Lateral Movement Due to Roll and Jounce** library (24). When doing so, be sure to disable this control (select **Not used** (25)), as shown in the figure.

A drop-down control specifies whether the function should not be used (recommended), use a linear coefficient (keyword = `SUSP_LAT_AXLE_JOUNCE_COEFFICIENT`), or a blue link to a **Suspension: Lateral Movement Due to Jounce** dataset describing the axle lateral movement as a 1D function of axle jounce.

If a dataset is linked from the **Suspension: Lateral Movement Due to Roll and Jounce** library (24), then any settings made with this control (25) will override any settings from the linked dataset (24).

(26) Setting to specify axle steer as a Configurable Function of roll relative to axle and possibly axle jounce. Axle steer is positive when it produces positive steer as defined at the wheel (positive when the wheels turn to the left).

A drop-down control specifies whether the function should use a linear coefficient (keyword = `SUSP_AXLE_ROLL_STEER_COEFFICIENT`) as shown in the figure, or a blue link to a **Suspension: Roll Steer (Solid Axle)** dataset which supports 1D functions of axle roll, 2D functions of axle roll and axle jounce, or a combination of 1D functions (e.g., summing a 1D function of axle roll with a 1D function of axle jounce).

Axle steer applies in the plane of the rolled axle.

(27) Setting to an animation dataset that can be associated with the solid axle. This is most often used to link to an animator shape used to visualize the solid axle in animations.

The custom settings checkbox (11), described previously, enables a miscellaneous yellow field which may be particularly useful for advanced solid axle settings, Figure 14. In this example, two math model keywords are set to nonzero values:

1. Pitch moment of inertia of the axle body (keyword = `IA_YY`).
2. X coordinate of the center of gravity for the axle body (`X_CG_AXLE`). This is relative to the longitudinal center of the suspension, which is itself located with X coordinate `LX_AXLE` in the sprung mass coordinate system, from the sprung mass screen.

The two math model keywords above are indexed by axle. When specifying from the custom settings area on the **Suspension: Solid Axle Kinematics** screen, it is not necessary to include the axle index.

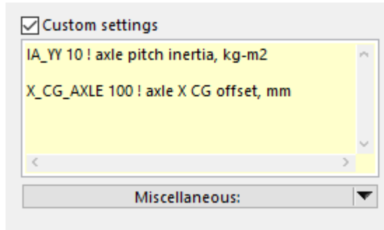


Figure 14. Custom settings field used for additional solid axle inertial properties.

CarSim Virtual Steering Axis System Kinematics

This screen is not available in TruckSim.

The *virtual steering axis* suspension is often encountered on the front axle of luxury coupes and sedans. It can be thought of as an independent suspension linkage where one or both control arms (wishbones) have been split into a pair of tension-compression links. As such, the lower or upper ball joint is then replaced by *double ball joints*. With the virtual steering axis, the steer of the roadwheel relative to the sprung mass is no longer given by a fixed axis represented by parameters such as kingpin inclination or caster. Instead, the steer of the roadwheel relative to the sprung mass is about an instantaneous screw axis implied by the suspension linkage's geometry. In CarSim, beginning in version 2020.1, this suspension type is modeled using suspension kinematics functions depending on both jounce and steering rack travel. These values are based on exercising a detailed multibody model, such as one from SuspensionSim, or a physical vehicle on a K&C rig by jouncing the wheel for various rack travel settings.

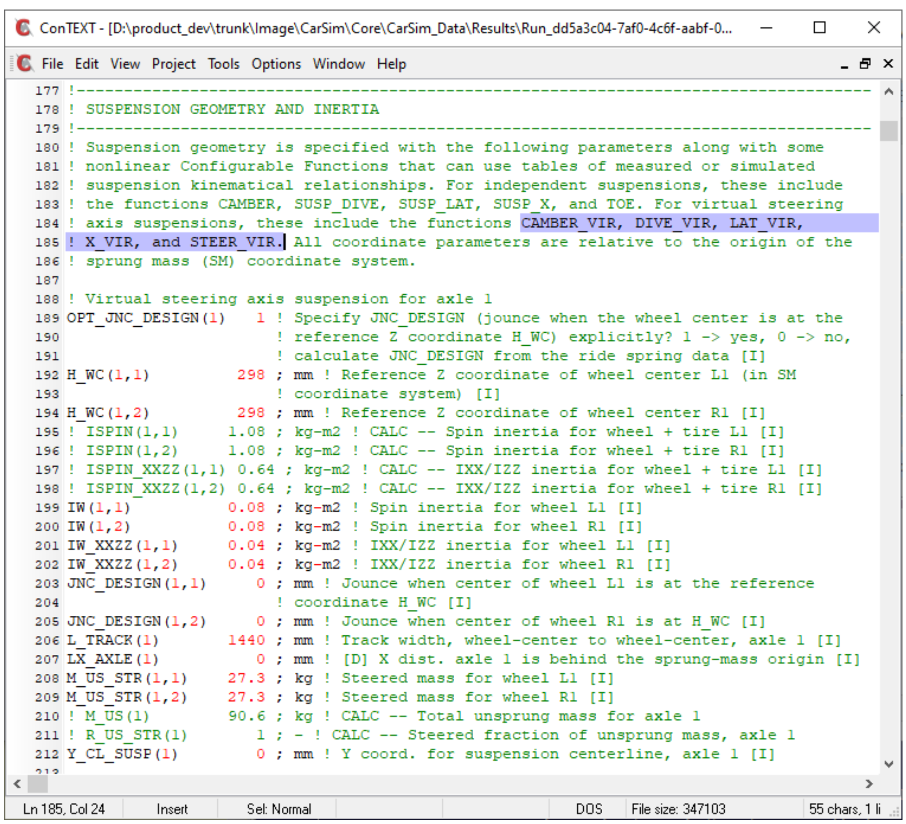


Figure 15. Echo file portion for the virtual steering axis suspension type.

Figure 15 shows an Echo file portion for a run using the virtual steering axis type. Compared to the independent suspension (Figure 9, p. 12), there are fewer parameters, considering there is no distinction between steered and non-steered parts of the unsprung mass, and there is a different method for handling static camber and toe. The complexity instead moves to the required 2D kinematics tables, as covered below.

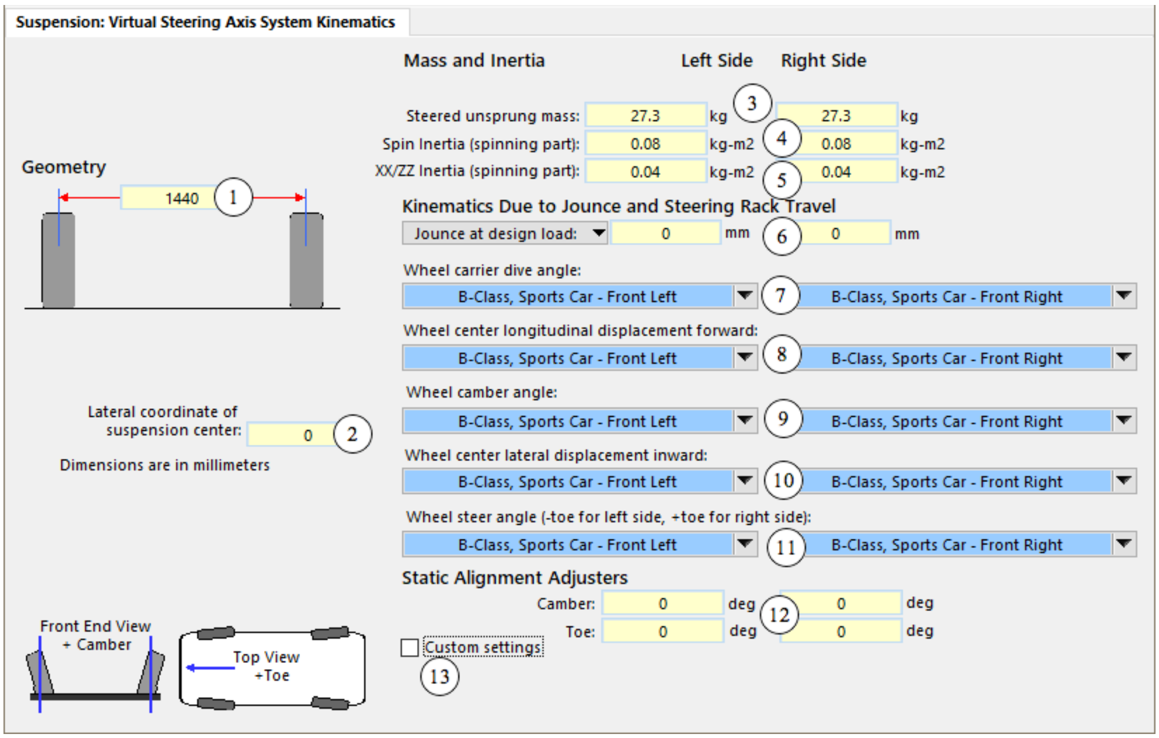


Figure 16. The Virtual Steering Axis Kinematics screen, available in CarSim.

The virtual steering axis kinematics screen is shown in Figure 16. The wheel kinematics definitions are covered in this section, with additional coverage in the later section Suspension Kinematics in the Math Models, p. 30.

1 Track width (keyword = `L_TRACK`) when the wheel centers are at the reference heights. This is between the wheel centers, not between the centers of tire contact.

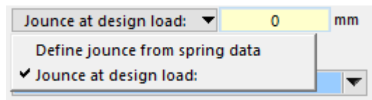
2 Lateral coordinate of the suspension center (keyword = `Y_CL_SUSP`). Normally zero, this can be given a non-zero value if the wheels are not located symmetrically about the longitudinal centerline of the vehicle sprung mass.

3 Steered unsprung mass on one side (keyword = `M_US_STR`). This mass includes the parts of the suspension that are steered, not including the mass of the tire and rim that is specified on the **Tire** screen. This is the same as the suspension unsprung mass (not including tire and rim), because all suspension unsprung mass is steered with the virtual steering axis suspension.

4 Spin moment of inertia for non-removable parts of the wheel (keyword = `IW`). The total spin inertia for each wheel is the sum of this value plus the inertia due to the tire, as specified on the **Tire** screen. The total spin inertias for each side (`ISPIN`) are shown in the suspension geometry and inertia section of the Echo file.

(5) Yaw and roll moment of inertia for non-removable parts of the wheel (keyword = `IW_XXZZ`). The same value is applied to both the yaw (Izz) and roll (Ixx) moments of inertia for the wheel. The total yaw/roll moment inertia for each wheel is the sum of this value plus the inertia due to the tire, as specified on the **Tire** screen. The total inertias for each side (`ISPIN_XXZZ`) are shown in the suspension geometry and inertia section of the Echo file.

(6) Pull-down control for choosing between two jounce definitions:



When the first option is chosen, the jounce at the design load condition is defined as the jounce in the spring table that corresponds to the spring force needed to support the sprung mass in the design load condition.

When the second option is chosen, an adjacent data field is displayed in which you specify the jounce when the vehicle is in the design load condition (keyword = `OPT_JNC_DESIGN`).

The first option links the jounce definition to the spring datasets. It is convenient if you do not change those datasets much. In this case, you can change the sprung mass properties without worrying about the nonlinear effects of jounce.

The second option links the jounce definition to the wheel-center design height (`H_WC`). It is convenient if you do not change the design height much. In this case, you can change spring properties without making corresponding adjustments to the toe and camber tables.

Recall that an earlier section discussed some of the considerations in choosing between these two options (page 9).

(7) Blue link to define *wheel carrier dive angle* as a function of jounce and steering rack travel. The wheel carrier (knuckle) is mostly oriented by camber and steer, which establish the wheel spin axis and thus the spindle direction. Dive angle refers to the rotation of the wheel carrier about the oriented spin axis which is necessary to establish the true spatial orientation of the wheel carrier. For outboard-braked wheels (standard in CarSim), dive is equivalent to wheel spin angle, provided the brakes are locked during the K&C tests. Dive angle is positive if the wheel spins as if it is rolling forward.

(8) Blue link to define *wheel center longitudinal displacement* as a function of jounce and steering rack travel. This is the displacement of the wheel center in the sprung mass +X direction.

(9) Blue link to define *wheel camber angle* as a function of jounce and steering rack travel. Camber is defined here as the angle between the wheel plane and the sprung mass Z axis, positive when the top of the wheel leans away from the sprung mass, as is typical practice.

(10) Blue link to define *wheel center lateral displacement* as a function of jounce and steering rack travel. This is the displacement of the wheel center in the inward direction (sprung mass +Y direction for right wheel, sprung mass -Y direction for left wheel).

(11) Blue link to define *wheel steer angle* as a function of jounce and steering rack travel. Steer is defined as the angle between the sprung mass X axis and the line of intersection of the wheel plane and the nominal road surface (a surface parallel to the sprung mass X-Y plane). Steer is

positive when a wheel is steered to the left. For a right wheel, toe is equal to steer; for a left wheel, toe is equal to the negative of steer.

(12) Yellow fields to adjust static wheel alignment (static camber and toe). These set offsets in the 2D camber (`CAMBER_VIR`) and 2D steer (`STEER_VIR`) tables, respectively.

For camber, the static camber adjustment value is directly used to set `CAMBER_VIR_OFFSET`. The true static camber angle is then the sum of this camber offset and the value in the table corresponding to static equilibrium.

For toe, the static toe adjustment value is applied directly for the right wheel to `STEER_VIR_OFFSET`. For the left wheel, the static toe adjustment value is converted to a steer adjustment by multiplying by -1, then applied to `STEER_VIR_OFFSET`. This is because toe is equal to steer for the right wheel, while toe is equal to negative steer for the left wheel. The true static toe is then the sum of the steer offsets plus the steer value in the 2D table corresponding to static equilibrium.

These adjustment values are convenient for quick, small changes. Users who have a detailed multibody suspension model available, such as from SuspensionSim, may instead prefer to adjust the alignment in the detailed multibody model using its physical adjusters (eccentrics, shims, tie rods, etc.) and then regenerate the 2D suspension kinematics tables.

(13) Checkbox for custom settings. Use to display two miscellaneous blue links. These controls have no predefined purpose, but they can be used to specify parameters for extensions to the model or information for the animator. They are not displayed unless the box is checked.

Twist Beam System Kinematics

Twist beam or *twist axle* suspensions are frequently used at the rear of small front-wheel-drive passenger cars. They have simple construction, light weight, low cost, and occupy little space. In general, twist axles consist of a pair of trailing arms tied together by a lateral tube, channel, or beam. The beam provides lateral location of the trailing arms and contributes roll stiffness as it twists when the vehicle rolls. Jounce stiffness and damping are provided by a pair of coil-over spring and damper units, separate springs and dampers, or torsion bars and dampers.

The **Suspension: Independent System Kinematics** screen supports 2D Configurable Functions that represent wheel movements due to changes in jounce from both sides. For example, inclination of the left wheel is influenced by the Configurable Function `CAMBER` that calculates camber from the jounce from the left (this side) and jounce on the right (other side). The Technical Memo *Twist Beam Suspensions: Using 2D Tables* describes how to use measurable K&C data to simulate a twist beam suspension in CarSim and TruckSim.

CarSim also includes two sets of legacy suspension screens with alternative models for twist beam suspensions:

1. A custom set of CarSim screens uses the generic/independent suspension model and adds VS Commands to set compliance coefficients. This option is named Twist Beam 2016, because it was introduced in version 2016.0. This is covered in the Technical Memo: *Twist Beam Suspensions: Using VS Commands*.

2. A custom suspension model based on a solid axle was used in CarSim in versions prior to 2016.0 and is named Twist Beam (Legacy).

Twist beams are available only at the rear suspension, only on vehicles with independent front suspension, and are never actively steered. (Steering is due only to roll steer kinematics and suspension compliance.)

CarSim Twist Beam Kinematics Screen (2016)

This screen is not available in TruckSim.

The **Twist Beam System Kinematics** screen (Figure 17) is nearly the same as the **Independent System Kinematics** screen described earlier (Figure 10, page 13), with the difference being that a few of the controls were removed. The VS Commands used to define properties for a twist beam suspension are written by the **Twist Beam System Compliance** screen, described later.

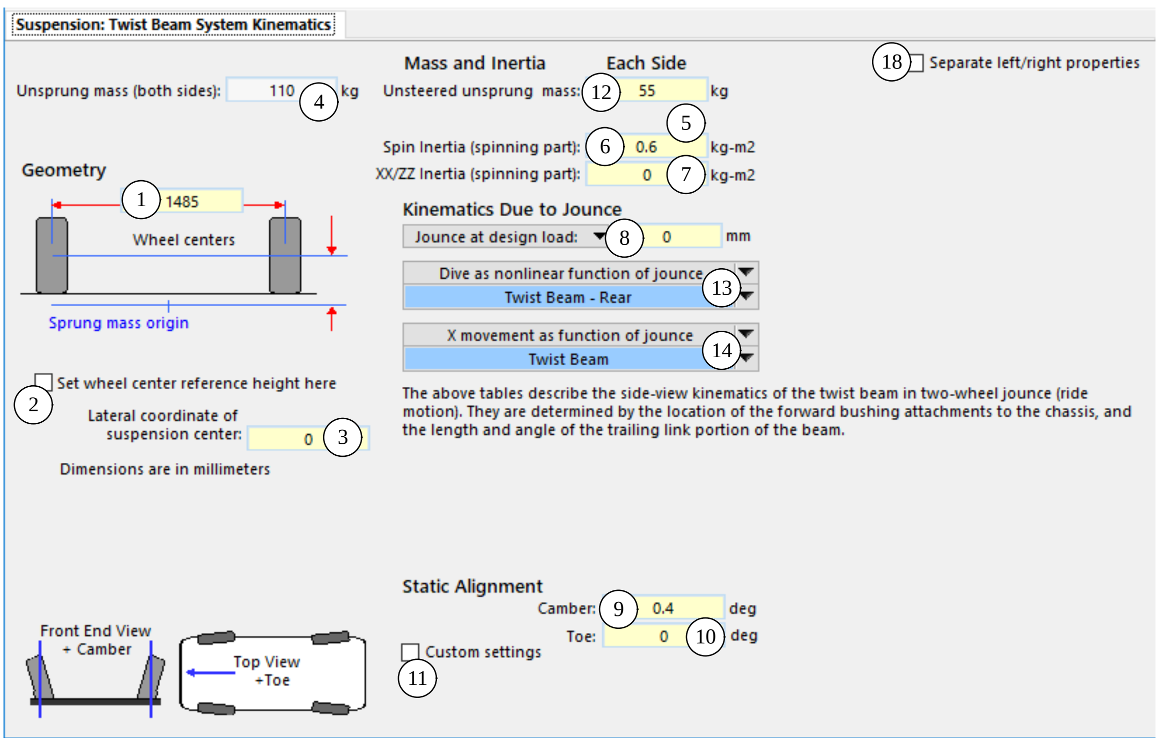


Figure 17. The Suspension: Twist Beam Kinematics screen.

CarSim Suspension: Twist Beam Kinematics (Legacy) screen

This screen is not available in TruckSim.

The oldest option for using a twist-beam suspension in CarSim uses a distinct suspension model with the layout code Twist, and distinct screens for kinematics and compliance.

In a twist-beam suspension, lateral forces are transmitted to the sprung mass through lateral reactions at bushings attaching the structure to the chassis, and through vertical reactions at the same bushings caused by the structural deformation (*twist*) of the beam. Complete characterization of load transfer with this type of suspension requires information about the structural flexibility of the cross-beam, especially the location of a point called the *shear center*, which, along with the forward bushing locations, determines the roll center. The shear center location is not easily measured by typical suspension tests, so this model uses a roll center to complete the description of suspension load transfer properties.

There are two special cases of the twist beam. The first has the lateral beam at the axis of the bushings attaching the beam to the chassis. This suspension really operates as a pure trailing arm with an anti-roll bar and should be modeled using the generic/independent suspension or the 2016 option. The second has the beam in the plane of the wheel centers, with a lateral locating link such as a track bar. This configuration may be modeled with the legacy Twist Beam model. Figure 18 shows the **Twist Beam Kinematics (Legacy)** screen. It is similar to the **Independent System Kinematics** and **Twist Beam System Kinematics** screen, but includes two additional controls.

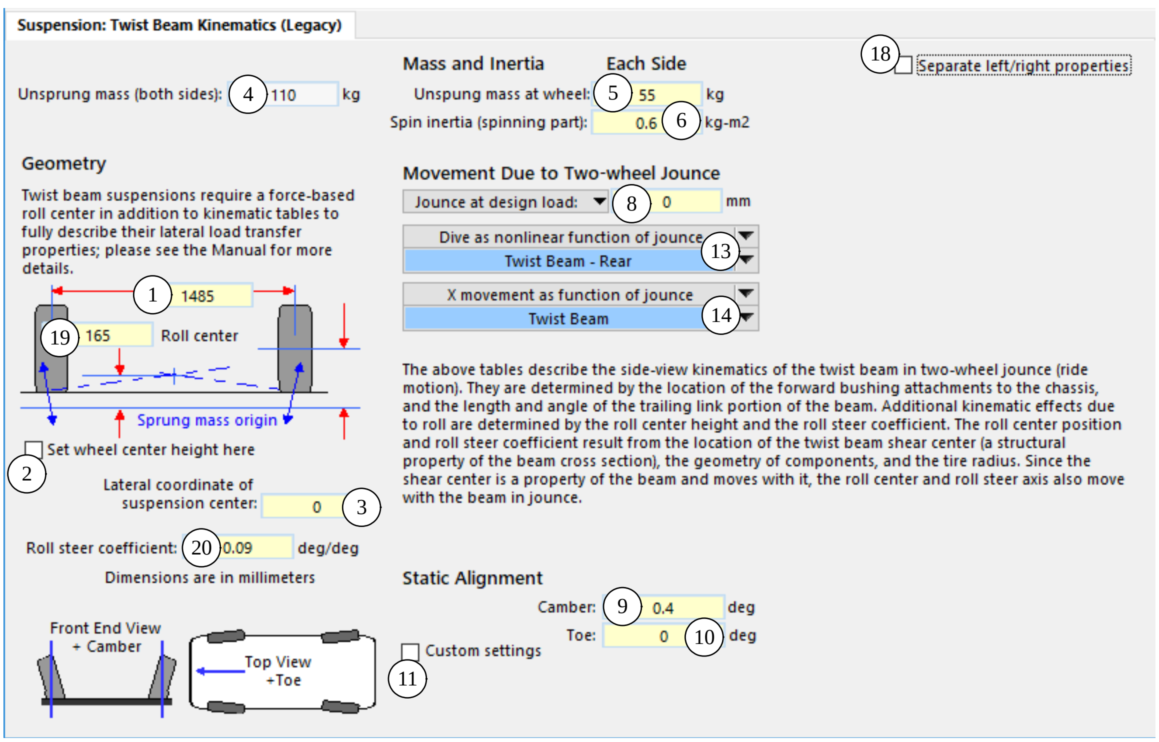


Figure 18. The Suspension: Twist Beam Kinematics (Legacy) screen.

(19) Height of axle roll center above the sprung mass origin at the design load condition (keyword = `H_RC`).

(20) Roll Steer Coefficient (keyword = `R_ROLL_STEER`). Ratio of the degrees of axle steer (positive to steer left about the Z axis) per degree of body roll (positive when rolled to the right about the X axis) relative to the axle.

The roll center concept defines a point that is a property of the suspension, with a specific tire, at a specific load condition. Figure 19 shows a graphical layout of suspension kinematics for two sizes of tires, overlaid such that the heights of the wheel centers are the same for the two example sets of tires. Vertical movement of the right-hand wheel is defined by an instant center (on the left-hand side of the figure), whose location is in turn defined by suspension geometry and compliance details. Vertical movement of the left-hand wheel is defined by an instant center on the other side. The roll center is the intersection of lines from the tire contacts to the suspension instant centers.

Figure 19 shows that from the point of view of the wheel centers, the roll center for the small tires is above the roll center for the big tires. From the point of view of the ground, the roll center for the small tires is not as far above the ground as the roll center for the big tires. The exact relationship between the roll center location and the loaded tire radius depends on the lateral location of the suspension instant centers. In the case of a twist-beam suspension, the instant centers are not just due to suspension geometry, but also to compliance effects.

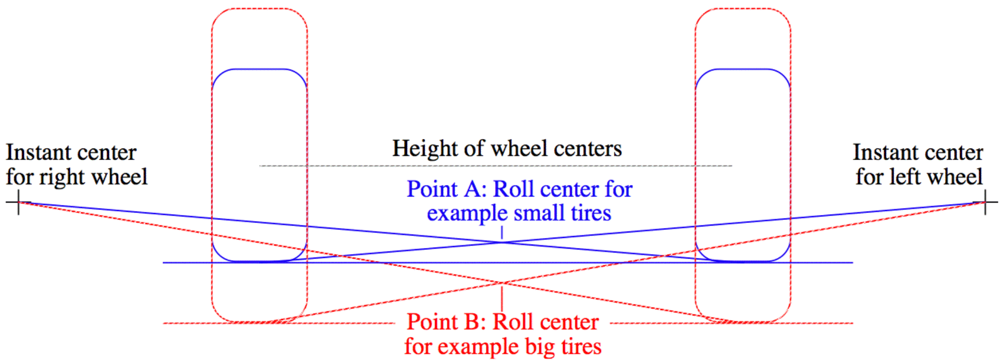


Figure 19. Effect of tire size on roll center location

These interactions depend on the tire size and design load configuration. This means that if you change the tire or the design load condition, you should determine the correct roll center position for the new vehicle configuration.

Configurable Functions for Suspension Kinematics

Table 1 lists the library screens provided to describe nonlinear kinematical relationships that are typically measured with a K&C test rig. Each screen has a Configurable Function with standard editing and viewing controls that are described in detail in the *VehicleSim Browser Reference Manual*. VS Math Models identify the Configurable Functions using keywords based on the root names listed in the table. These function names are also listed in the Suspension Geometry and Inertia section of the Echo file (e.g., Figure 9, page 12).

Recall that suspension module types are represented with the codes `Ind` (generic/independent), `SA` (solid axle), `Vir` (Virtual steering axis), and `Twist` (legacy Twist Beam). Table 1 uses these codes to identify which modules use the listed Configurable Functions.

As noted earlier, the generic/independent suspension model supports five 2D kinematics tables with additional independent variable that is the jounce of other side of suspension. Similarly, the solid axle model supports four 2D tables for combined effects of axle jounce and roll relative to axle. The virtual steering axis type supports five 2D kinematics tables with independent variables of jounce and steering rack travel.

Table 1. Summary of suspension kinematical table libraries.

Library Screen	Root Keyword	Suspension Types
Suspension: Camber Angle	`CAMBER`	Ind
Suspension: Lateral Movement Due to Jounce	`SUSP_LAT`	
Suspension: Toe Angle	`TOE`	
Suspension: Dive Angle (Caster Change)	`SUSP_DIVE`	Ind, Twist
Suspension: Longitudinal Movement	`SUSP_X`	
Suspension: Lateral Movement Due to Roll and Jounce	`SUSP_Y_AXLE_ROLL`	SA
Suspension: Roll Steer (Solid Axle)	`SUSP_AXLE_ROLL_STEER`	
Suspension: Longitudinal Movement (Solid Axle)	`SUSP_X_AXLE`	
Suspension: Dive Angle (Solid Axle)	`SUSP_DIVE_AXLE`	
Suspension: Camber Angle (Virtual Steering Axis)	`CAMBER_VIR`	Vir
Suspension: Dive Angle (Virtual Steering Axis)	`DIVE_VIR`	
Suspension: Lat. Movement (Virtual Steering Axis)	`LAT_VIR`	
Suspension: Long. Movement (Virtual Steering Axis)	`X_VIR`	
Suspension: Steer Angle (Virtual Steering Axis)	`STEER_VIR`	

Note While the **Suspension: Dive Angle (Caster Change)**, **Suspension: Lateral Movement Due to Jounce**, and **Suspension: Longitudinal Movement** libraries can be used with solid axle (SA) suspension types for legacy datasets, they are no longer recommended for solid axle suspensions. The recommended library screens for solid axles are given in Table 1. The recommended library screens support the 2D table modes. Using the non-solid-axle specific dive, lateral, and longitudinal movement screens in 2D mode will result in an error, as these use jounce-other-side as the second independent variable rather than axle jounce or axle roll.

Suspension Kinematics in the Math Models

In previous sections, the various suspension types have been described in terms of their kinematics inputs, with references to how particular inputs such as dive are applied to the various combinations of models and options. The generic/independent suspension model warrants further exposition in this regard. This section also includes descriptions of the suspension kinematics outputs useful when putting together new suspension datasets.

General Kinematics Outputs

The following wheel kinematics outputs are available for all suspension types and are intended to represent the measurements from a K&C test. <wheel> refers to a wheel-identifier like `L1`, `R2`, etc. This is not an exhaustive list of suspension kinematics outputs, rather, ones of particular interest when validating the wheel or axle kinematics.

- `Xrel_`<wheel>. X-coordinate of the wheel center in the sprung mass coordinate system.
- `Yrel_`<wheel>. Y-coordinate of the wheel center in the sprung mass coordinate system.

- `Zrel_`<wheel>. Z-coordinate of the wheel center in the sprung mass coordinate system.
- `DiveG_`<wheel>. The change in orientation of the wheel carrier about the wheel spin axis due to suspension kinematics and compliance. With locked outboard brakes, this is equivalent to the wheel spin change. When the wheel is viewed from the left side of the vehicle, this is positive counterclockwise.
- `Camber`<wheel>. The angle between the wheel plane and the sprung mass Z axis, with the usual sign convention for camber (top in = negative).
- `Steer_`<wheel>. The angle between the sprung mass X axis and the line of intersection of the wheel plane and nominal road surface. (The nominal road surface is parallel to the sprung mass X-Y plane.) This is positive if the wheel is steered to the left and corresponds to a rotation about the sprung mass Z axis.

The general outputs for dive, camber, and steer correspond to a 3-1-2 rotation of the wheel carrier relative to the sprung mass (steer-inclination-dive, where camber = inclination for a right wheel, camber = -inclination for a left wheel).

Generic/Independent Kinematics

Translations and Rotations in the Ind Model

An important feature of the generic/independent model, denoted by code `Ind` (GUI) or `I` (Echo file), is that steering takes place about an inclined and offset steering axis. The inclination and offset of the axis are defined in a reference frame by four parameters: kingpin inclination, caster angle, and lateral and longitudinal offset from the wheel center (measured at the height of the wheel center). This reference frame has the same axis directions as the sprung mass reference frame, so it is also necessary to specify the static camber and static toe corresponding to this kingpin geometry (see Kingpin Geometry and Spin Axis, p. 15).

The reference frame containing the steering axis has its position and orientation relative to the sprung mass defined with three sequential translations (Z, X, and Y) and two sequential rotations (roll and pitch). The final location and orientation of the wheel is established by rotating the wheel about the steering axis according to the total steer angle (with contributions from the toe table, steering system, compliance, etc.).

Tables of data used to characterize steer/toe are typically measured in K&C rigs in the ground plane, perpendicular to the vehicle Z axis (sprung mass). The model has long had an option (`OPT_STEER_DEF`) to convert steer angles from the vehicle Z axis to the steer axis, which is used internally in the model.

The jounce (Z) movement is unconstrained in the model. The four other degrees of freedom of the steering axis reference frame (X, Y, roll, pitch) are kinematically constrained to be functions of jounce, according to the suspension kinematics tables (longitudinal movement, lateral movement, dive, and camber, respectively). The steer/toe degree of freedom is a function of jounce for non-steered independent suspensions. Steered independent suspensions also include steering gear output (Pitman arm or rack travel) as a contribution to steer/toe. There are also compliance effects for the five constrained variables (X, Y, roll, pitch, steer/toe), based on applied tire forces and moments.

Because the translations and rotations used in the model are sequential, not all can be measured physically because they involve reference frames that do not correspond to physical parts. For example, the X, Y, and Z movements of the generic/independent model are often interpreted as that of the wheel center point. However, with toe changing due to jounce and typical steering axis geometry, the toe table implies a rotation about the steering axis which moves the wheel center into a different position than implied solely by the model's X, Y, or Z values.

Independent Suspension Kinematics Utility (IKU)

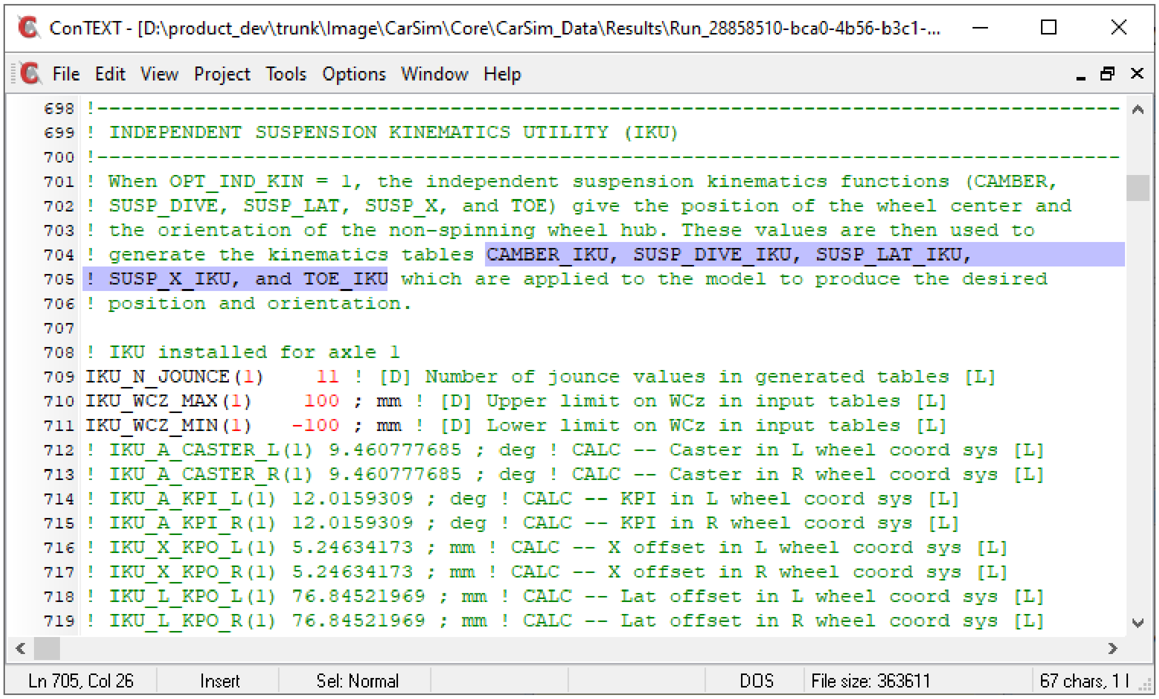
As of CarSim and TruckSim 2022.0, there is an option to interpret longitudinal movement, lateral movement, dive, and camber with definitions that more closely match physical measurements or general simulation outputs (SuspensionSim, ADAMS, those produced by CarSim or TruckSim, p. 30). This new option is like the older `OPT_STEER_DEF` option, wherein the data provided for toe is applied directly to the model (as a kingpin rotation) if the option is off or converted into the kingpin rotation that produces the desired toe about the vehicle Z axis if the option is on.

The 2022.0 option is called `OPT_IND_KIN`, and when it is on (=1), the VS Math Model calculates the independent suspension model's X, Y, roll, and pitch motions such that the longitudinal movement, lateral movement, dive, and camber inputs apply to the wheel position and wheel orientation, rather than that of the steering axis reference frame. (Specifically, the option uses a 3-1-2 rotation of the wheel, followed by the translation.) The `OPT_STEER_DEF` option being on (=1) is a prerequisite for turning on `OPT_IND_KIN`.

Specifically, turning on `OPT_IND_KIN` installs the Independent Suspension Kinematics Utility (IKU), as shown in Figure 20. The input tables for camber, dive, lateral movement, longitudinal movement, and toe — `CAMBER`, `SUSP_DIVE`, `SUSP_LAT`, `SUSP_X`, and `TOE`, respectively — are all then interpreted as applying to the wheel's position and orientation, rather than the kingpin and the rotation about the kingpin. The IKU then uses these input data to generate new kinematics tables — `CAMBER_IKU`, `SUSP_DIVE_IKU`, `SUSP_LAT_IKU`, `SUSP_X_IKU`, and `TOE_IKU` — which are applied to the model to produce the desired wheel kinematics.

The dimensions and extent of the generated tables are established by three parameters, as seen in lines 709-711 of the Echo file, Figure 20. The generated tables are always square tables of the `2D_SPLINE` type (number of rows = number of columns = `IKU_N_JOUNCE`). The other two parameters, `IKU_WCZ_MAX` and `IKU_WCZ_MIN`, instruct the IKU what the upper and lower limits are on the wheel center Z displacement in the input tables. This is required because some input types, such as `COEFFICIENT`, do not have clear limits.

There are no GUI fields for the three IKU parameters associated with each installed IKU. These are indexed by axle and can be set in a miscellaneous yellow field if the defaults are not appropriate. If they are set using a Generic VS Commands dataset linked to the custom settings area (item ⑪, Figure 10, p. 13), it is not necessary to explicitly include the indexes.


ConTEXT - [D:\product_dev\trunk\Image\CarSim\Core\CarSim_Data\Results\Run_28858510-bca0-4b56-b3c1-...

File Edit View Project Tools Options Window Help

```
698 !-------------------------------------------------------------------------------
699 ! INDEPENDENT SUSPENSION KINEMATICS UTILITY (IKU)
700 !-------------------------------------------------------------------------------
701 ! When OPT_IND_KIN = 1, the independent suspension kinematics functions (CAMBER,
702 ! SUSP_DIVE, SUSP_LAT, SUSP_X, and TOE) give the position of the wheel center and
703 ! the orientation of the non-spinning wheel hub. These values are then used to
704 ! generate the kinematics tables CAMBER_IKU, SUSP_DIVE_IKU, SUSP_LAT_IKU,
705 ! SUSP_X_IKU, and TOE_IKU which are applied to the model to produce the desired
706 ! position and orientation.
707
708 ! IKU installed for axle 1
709 IKU_N_JOUNCE(1)     11 ! [D] Number of jounce values in generated tables [L]
710 IKU_WCZ_MAX(1)     100 ; mm ! [D] Upper limit on WCz in input tables [L]
711 IKU_WCZ_MIN(1)    -100 ; mm ! [D] Lower limit on WCz in input tables [L]
712 ! IKU_A_CASTER_L(1) 9.460777685 ; deg ! CALC -- Caster in L wheel coord sys [L]
713 ! IKU_A_CASTER_R(1) 9.460777685 ; deg ! CALC -- Caster in R wheel coord sys [L]
714 ! IKU_A_KPI_L(1) 12.0159309 ; deg ! CALC -- KPI in L wheel coord sys [L]
715 ! IKU_A_KPI_R(1) 12.0159309 ; deg ! CALC -- KPI in R wheel coord sys [L]
716 ! IKU_X_KPO_L(1) 5.24634173 ; mm ! CALC -- X offset in L wheel coord sys [L]
717 ! IKU_X_KPO_R(1) 5.24634173 ; mm ! CALC -- X offset in R wheel coord sys [L]
718 ! IKU_L_KPO_L(1) 76.84521969 ; mm ! CALC -- Lat offset in L wheel coord sys [L]
719 ! IKU_L_KPO_R(1) 76.84521969 ; mm ! CALC -- Lat offset in R wheel coord sys [L]
```

Ln 705, Col 26 | Insert | Sel: Normal | DOS | File size: 363611 | 67 chars, 1 l

Figure 20. Echo file portion showing the Independent Suspension Kinematics Utility installation.

> **Note** Because OPT_IND_KIN=1 requires that OPT_STEER_DEF=1, the TOE_IKU table generated by the IKU is very similar to the toe input table TOE. The difference is that (a) the TOE table's independent variables will be interpreted as wheel center Z displacements, rather than independent suspension model Z's, and (b) the TOE_IKU table will always be the 2D_SPLINE type, consistent with the dimensions and extent established by the IKU parameters.

The IKU accounts for the relationship between the spin axis and the steering axis as established by the static camber and static toe values (A_CAMBER and A_TOE, respectively). In doing so, the IKU calculates the kingpin geometry corresponding to an axis system where the spin axis is aligned with the Y axis. In this axis system, the caster, kingpin inclination, X offset, and lateral offset will all differ slightly from the input values, provided static camber and static toe are nonzero. The kingpin geometry parameters in this alternative axis system are echoed as read-only parameters (lines 712-719, Figure 20) and may be of interest to someone working to understand the effect of the static camber and static toe parameters within CarSim and TruckSim.

Generic/Independent Kinematics Outputs

Outputs are included which reflect the model's X, Y, Z, pitch, roll, and steer, rather than the general measurements covered previously. <wheel> refers to a wheel-identifier like L1, R2, etc.

- SusX_<wheel>. X-displacement of the kingpin reference frame in the sprung mass coordinate system. Combined effects of longitudinal kinematics (SUSP_X if OPT_IND_KIN off, SUSP_X_IKU if OPT_IND_KIN on) and compliance (C_LONG).

- SusY_<wheel>. Y-displacement of the kingpin reference frame in the sprung mass coordinate system, positive if the wheel moves inward. Combined effects of lateral kinematics (SUSP_LAT if OPT_IND_KIN off, SUSP_LAT_IKU if OPT_IND_KIN on) and compliance (C_LAT).
- Jnc_<wheel>. Z-displacement of the kingpin reference frame in the sprung mass coordinate system. This is the primary independent variable of the independent suspension, with the secondary independent variable being the jounce of the other wheel on the axle.
- Dive_<wheel>. The change in orientation of the kingpin reference frame corresponding to (but not identical to) caster change, positive counterclockwise when viewed from the left side of the vehicle. Combined effects of dive kinematics (SUSP_DIVE if OPT_IND_KIN off, SUSP_DIVE_IKU if OPT_IND_KIN on) and compliance (CD_MY).
- CamUS<wheel>. The change in orientation of the kingpin reference frame corresponding to kingpin inclination change, using the sign convention for camber. Combined effects of camber kinematics (CAMBER if OPT_IND_KIN off, CAMBER_IKU if OPT_IND_KIN on), camber compliance (CC_FX), and inclination compliance (CI_MZ, CI_FY).
- StrKp_<wheel>. The rotation about the inclined and offset steering axis (kingpin). How the input data applies to this depends on the options selected:
    - When OPT_STEER_DEF is off, this directly represents the combined effects of toe kinematics (TOE) and compliance (CT_FX), and steer kinematics (RACK_KIN or STEER_KIN) and compliance (CS_FY, CS_MZ, STEER_COMP).
    - When OPT_STEER_DEF is on, the combined effects of toe kinematics (TOE if OPT_IND_KIN off, TOE_IKU if OPT_IND_KIN on) and compliance (CT_FX), and steer kinematics (RACK_KIN or STEER_KIN) and compliance (CS_FY, CS_MZ, STEER_COMP) are interpreted as being about the sprung mass Z axis, and the total is converted into the kingpin rotation which produces that total steer about the Z axis.

Solid Axle Kinematics Outputs

Outputs are included which represent the model's translations and rotations, rather than the general measurements covered previously. <axle> refers to an axle identification number like 1, 2, etc.

- SusX_A<axle>. X-displacement of the solid axle center in the sprung mass coordinate system. Combined effects of longitudinal kinematics (SUSP_X_AXLE) and compliance (C_LONG_AXLE).
- SusY_A<axle>. Y-displacement of the solid axle center point in the sprung mass coordinate system assuming zero axle jounce. When the axle is jounced, there is an additional lateral position change of the center of the axle due to axle jounce being applied in the rolled Z direction. Provided the checkbox on the solid axle lateral movement screen is unchecked (see Figure 13 and surrounding discussion on p. 21), this output will match the combined effects of lateral kinematics (SUSP_Y_AXLE_ROLL) and compliance (C_LAT_AXLE). If the box is checked, the generic output Yrel_A<axle>, the axle center Y-coordinate in sprung mass coordinates, will be more relevant.
- JounceA<axle>. Z-displacement of the axle center point in the rolled axle's Z direction. This is one of the independent variables for the solid axle model.
- Dive_A<axle>. The dive rotation of the axle body corresponding to dive kinematics (SUSP_DIVE_AXLE).

- Roll_A<axle>r. Axle roll, also known as the roll relative to axle, positive when the right wheel moves up and the left wheel moves down. This is one of the independent variables for the solid axle model.
- StrR_A<axle>. Steer change of the axle body about the rolled Z axis, corresponding to roll steer kinematics (SUSP_AXLE_ROLL_STEER).

Virtual Steering Axis Kinematics Outputs (CarSim)

Outputs are included which represent the model's translations and rotations, rather than the general measurements covered previously. <wheel> refers to a wheel-identifier like L1, R1, etc.

- SusX_<wheel>. X-displacement of the wheel center, sprung mass coordinates. Combined effects of longitudinal kinematics (X_VIR) and compliance (C_LONG).
- SusY_<wheel>. Y-displacement of the wheel center, sprung mass coordinates, positive when the wheel moves inward. Combined effects of lateral kinematics (LAT_VIR) and compliance (C_LAT).
- Jnc_<wheel>. Z-displacement of the wheel center, sprung mass coordinates. This is one of the independent variables in the virtual steering axis model.
- Dive_<wheel>. Equivalent to general dive measurement DiveG_<wheel> and corresponding to the combined effects of dive kinematics (DIVE_VIR) and compliance (CD_MY).
- Camber<wheel>. General camber measurement Camber<wheel> here corresponds to the combined effects of camber kinematics (CAMBER_VIR) and compliance (CC_FX), and inclination compliance (CI_FY, CI_MZ).
- Steer_<wheel>. General steer measurement Steer_<wheel> here corresponds to the combined effects of steer kinematics (STEER_VIR) and compliance (CS_FY, CS_MZ, STEER_COMP), and toe compliance (CT_FX).
- Y_Rack<axle>. The rack travel, positive leftward, which is an independent variable in the virtual steering axis model.

Suspension Compliance in the Math Models

Each suspension in CarSim and TruckSim has compliances that cause small displacements in response to forces and moments applied by the ground through the tires where they contact the ground. These compliances relate steer, camber, lateral displacement, and longitudinal displacement of the wheels due to the deflection of bushings and other suspension components to tire forces and moments.

Testing of suspension compliance is typically done at zero steer angle. For steerable suspensions, the steer angle is set to zero for the test. Physically, the suspension compliances involving camber, lateral displacement, and longitudinal displacement mainly involve bushing and structural members that are not steered. Accordingly, the math model determines all compliance effects using the directions of the sprung mass X-Y-Z axes, as normally apply in a K&C test.

The math models always transform tire forces and moments from the ground to equivalent force and moment vectors applied at the wheel center. (This is done to support third-party tire models that produce forces and moments at the wheel center.) To calculate the compliance effects, the components of the force vector are taken in the sprung mass X, Y, and Z directions. The Z component of the moment is also taken.

Quasi-Static Compliance Calculations

As described earlier, most of the VS Suspension models have jounce and/or roll as dynamic variables that are used to define kinematical constraints and spring and damper forces. Suspension jounce and roll are state variables in the VS Math Model that are calculated by integrating ordinary differential equations (ODEs). As noted earlier, jounce displacements have ranges of motion of at least 10 cm, and usually higher. They contribute modes of vibration with frequencies in the range of 10 – 20 Hz.

On the other hand, the compliance deflections are much smaller, covering a few millimeters or less, or angles of a fraction of a degree. The dynamics of these deflections are usually negligible in affecting the system response of the vehicle to driver and ADAS controls, which are generally under 20 Hz. In the VS Math Models, the compliance deflections are handled quasi-statically. By "quasi-statically," we mean "nearly instantly."

At the start of each time step, forces from the tires in the previous time step are used to calculate the compliance deflections, and those are added to the relative translational coordinates or relative rotational coordinates. For example, the steer of a wheel at the road has the effects of all current kinematical relationships, plus all compliance effects based on forces that existed at the previous time step.

Generic/independent and solid axle compliances

For the generic/independent and solid axle suspension types, the following quasi-static adjustments are made:

- The force and moment components are used in Configurable Functions to obtain relative roll and pitch deflections for an un-steered body in the math model containing the steering (kingpin) axis (e.g., the steering knuckle).
- The steer deflection is added to the steering angle about the Z axis of the sprung mass (`OPT_STEER_DEF` on) or to the steering angle about the kingpin axis (`OPT_STEER_DEF` off).
- The Fx and Fy applied forces are in the directions of the sprung mass X and Y axes, regardless of how the wheel is steered.
- Although the rotation of the wheel about the X axis due to applied forces and moments matches the definition of *camber* during the test, the "camber compliance" is not applied to true camber when wheel steer angle is not zero. The small compliance rotation about the sprung mass X axis applies to the un-steered body that contains the steering (kingpin) axis, rather than the steered wheel.

CarSim virtual steering axis suspension compliances

For the CarSim virtual steering axis suspension type, the wheel-center-equivalent force and moment vectors, expressed in sprung mass coordinates as F_x, F_y, F_z, M_x, M_y, and M_z, apply to the suspension kinematics variables as follows:

- *Wheel carrier dive angle* is affected by M_y via compliance coefficient CD_MY [deg/(N-m)].
- *Wheel center longitudinal displacement* is affected by F_x via compliance coefficient C_LONG [mm/N].
- *Wheel camber angle* is affected by
  - F_x via camber compliance coefficient CC_FX [deg/N].
  - F_y via inclination compliance coefficient CI_FY [deg/N]. Inclination is equal to camber for a right wheel and equal to negative camber for a left wheel.
  - M_z via inclination compliance coefficient CI_MZ [deg/(N-m)].
- *Wheel center lateral displacement* is affected by F_y via compliance coefficient C_LAT [mm/N].
- *Wheel steer angle* is affected by
  - F_x via toe compliance coefficient CT_FX [deg/N]. Toe is equal to steer for a right wheel and equal to negative steer for a left wheel.
  - F_y via steer compliance coefficient CS_FY [deg/N].
  - M_z via steer compliance coefficient CS_MZ [deg/(N-m)].
  - The sum of M_z (left and right wheels) is also used with steer compliance coefficient COMP_STEER [deg/(N-m)] to modify wheel steer angle.

For the virtual steering axis, while the tire forces and moments are always with respect to the sprung mass axis system, the dive, camber, and steer effects are applied to the true kinematics variables. For example, CC_FX will cause a true camber change, modifying the angle between the wheel plane and the sprung mass Z in response to a force at the wheel center in the sprung mass X direction. Advanced users may be interested in using custom equations for these compliance coefficients to establish effects based on both jounce and steering rack travel. Defining the forces and moments in the sprung mass axis system is convenient for measuring such dependencies during a K&C test (e.g., set a jounce and rack position, apply a sprung mass X direction force at the wheel center, and measure resultant camber angle).

Dynamic X compliance for independent suspensions

This feature is only available in CarSim.

CarSim includes an alternate independent suspension module that has an extra dynamical DOF on each side for longitudinal movement relative to the sprung mass. This feature has been requested by some users involved with rapid braking, in which a dynamic modeling of the longitudinal compliance is preferred to the quasi-static compliance used in the normal independent suspension modules.

This module can be activated using a checkbox on the **Suspension: Compliance (Nonlinear)** screen, p. 71. It can also be added with a VS Command with the form:

DEFINE_SUSP_X_DOF <iaxle>.

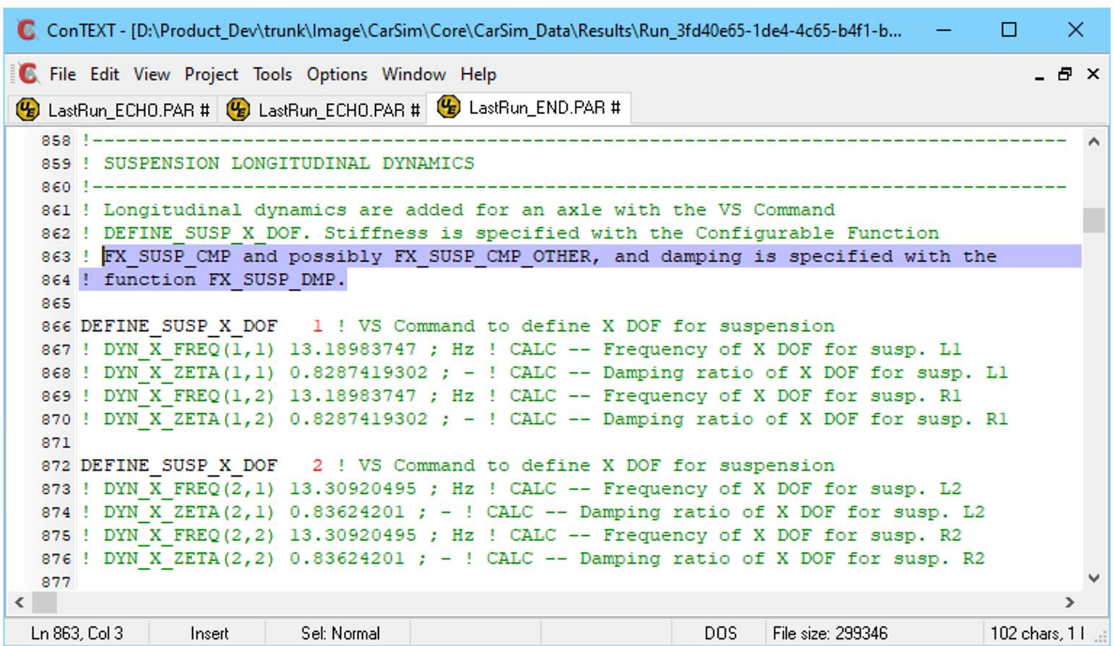
The dynamic module removes a constraint in the relative longitudinal displacement, making the displacement another ODE state variable. When this option is enabled, the Echo file will contain an additional section for each axle in which the command was applied (Figure 21).

The alternative module adds a spring and damper between the left suspension and the sprung mass and another spring and damper between the right suspension and the sprung mass. Overall, two

DOFs are added to the model. The spring applies a force using a 2D Configurable Function `FX_SUSP_CMP` that calculates a force as a function of the X displacement on “this side” and also the “other side.” This implies a secondary stiffness which establishes the amount of deflection on the other side given a force applied to the current side. This may be useful for representing the coupling between the left and right sides, perhaps from the subframe bushings or the anti-roll bar. As with most 2D Configurable Functions, it can be split to use a sum or product of two 1D functions, where the second function is `FX_SUSP_CMP_OTHER`.

Damping force is provided with a 1D Configurable Function `FX_SUSP_DMP` using the displacement rate on the same side as the force.

For convenience, the VS Math Model calculates an equivalent natural frequency `DYN_X_FREQ`, which is shown as read-only in the Echo file. The VS Math Model also calculates an equivalent damping ratio `DYN_X_ZETA`, which is shown in the Echo file.




```
!------------------------------------------------------------------------------------
! SUSPENSION LONGITUDINAL DYNAMICS
!------------------------------------------------------------------------------------
! Longitudinal dynamics are added for an axle with the VS Command
! DEFINE_SUSP_X_DOF. Stiffness is specified with the Configurable Function
! FX_SUSP_CMP and possibly FX_SUSP_CMP_OTHER, and damping is specified with the
! function FX_SUSP_DMP.

DEFINE_SUSP_X_DOF   1 ! VS Command to define X DOF for suspension
! DYN_X_FREQ(1,1) 13.18983747 ; Hz ! CALC -- Frequency of X DOF for susp. L1
! DYN_X_ZETA(1,1) 0.8287419302 ; - ! CALC -- Damping ratio of X DOF for susp. L1
! DYN_X_FREQ(1,2) 13.18983747 ; Hz ! CALC -- Frequency of X DOF for susp. R1
! DYN_X_ZETA(1,2) 0.8287419302 ; - ! CALC -- Damping ratio of X DOF for susp. R1

DEFINE_SUSP_X_DOF   2 ! VS Command to define X DOF for suspension
! DYN_X_FREQ(2,1) 13.30920495 ; Hz ! CALC -- Frequency of X DOF for susp. L2
! DYN_X_ZETA(2,1) 0.83624201 ; - ! CALC -- Damping ratio of X DOF for susp. L2
! DYN_X_FREQ(2,2) 13.30920495 ; Hz ! CALC -- Frequency of X DOF for susp. R2
! DYN_X_ZETA(2,2) 0.83624201 ; - ! CALC -- Damping ratio of X DOF for susp. R2
```

Figure 21. Portion of CarSim Echo file when dynamic X compliance is turned on for both axles.

The dynamic displacement of the suspension assembly relative to the sprung mass is given by the output variables `DefX_L1`, `DefX_R1`, etc. The rate of these deflections is given with the outputs `VxDef_L1`, etc. The total force from the spring and the damper is output with `Fx_C_L1`, etc.

If using the dynamic X degree of freedom, you should probably set the "Long. / Fx" compliance (math model keyword `C_LONG`) to zero so there is no need to separate the K&C measured compliance between the standard and dynamic effects. In this sense, the `DEFINE_SUSP_X_DOF` compliance replaces the standard X suspension compliance. The difference is it is a true dynamic effect which can vibrate and has damping, versus the quasi-static effect given by `C_LONG`. (If using the setting on the **Suspension: Compliance (Nonlinear)** screen, p. 71, `C_LONG` is replaced by `FX_SUSP_CMP` automatically.)

Screens for Suspension Compliance, Springs, and Dampers

The parameters and summary information for the suspension springs, dampers, and compliances are listed in a section of the Echo file (Figure 22). Each suspension in CarSim and TruckSim includes nine components that produce forces or moments based on suspension jounce and roll, relative to the sprung mass. All involve Configurable Functions, which are listed in the comments at the start of the Echo file section (lines 244-249, Figure 22). Six of these are springs (using functions `F_JNC_STOP`, `F_REB_STOP`, `FS_COMP`, and `FS_EXT` for three springs on each side), two are shock absorbers (using the function `FD` for a damper on each side), and one is an auxiliary roll moment defined by a spring (with function `MX_AUX`) and linear damper (coefficient = `DAUX`).

The suspension includes multiple compliance effects, also handled with Configurable Functions such as `CC_FX`, `CI_FY`, `CI_MZ`, `CS_FY`, `CS_MZ`, `CT_FX`, `CD_MY`, `C_LAT`, `C_LONG`, `C_LAT_AXLE`, and `C_LONG_AXLE`.

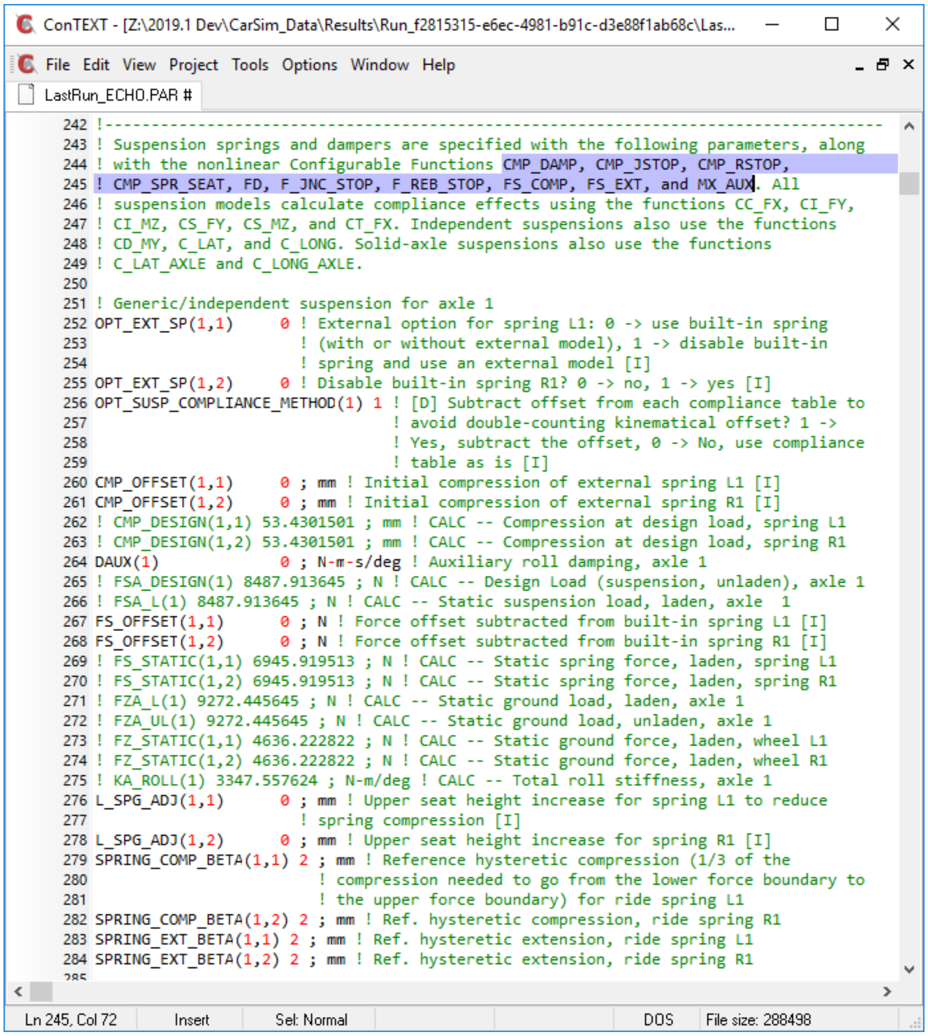
The parameters for a generic/independent suspension (e.g., axle 1 in Figure 22) mainly involve the details of how the forces and moments are calculated. The convention for an independent suspension is that the force actions are applied at the wheel center location, so the details of where the springs and dampers are physically located are not needed. Instead, Configurable Functions relate the jounce movements of the wheel center to compression of the force-producing elements. These functions are also listed in the Echo file: `CMP_DAMP`, `CMP_JSTOP`, `CMP_RSTOP`, and `CMP_SPR_SEAT` (see lines 244 and 245, Figure 22). These functions for damper, jounce stop, rebound stop, and spring compression, respectively, support 2D tables using independent variables of jounce and jounce on the other side.

In the case of the solid-axle suspension, the locations of the force-producing elements relative to the solid axle body may be included with additional parameters, as seen in the Echo file contents for a solid axle (Figure 23).

In most applications, the spacing distances between the elements on each side (`L_DAMPERS`, `L_SPRINGS`, etc.) are needed to properly account for the jounce force and roll moment between the axle and sprung mass.

> **Note** For the solid axle, there are multiple ways to enter the spring, damper, jounce stop, and rebound stop compression data. It is critical that the input parameters are consistent with the intended method to avoid the double counting of effects. For more on this, see *Solid Axle Spring and Damper Geometry*, beginning on p. 48, and *Solid Axle Jounce and Rebound Stop Geometry*, beginning on p. 51.

The virtual steering axis suspension (in CarSim only) uses the same compliance parameters as the generic/independent, except that the model keywords `CMP_DAMP_VIR`, `CMP_SPR_SEAT_VIR`, `CMP_JSTOP_VIR`, and `CMP_RSTOP_VIR` are used instead of `CMP_DAMP`, `CMP_SPR_SEAT`, `CMP_JSTOP`, and `CMP_RSTOP`. These allow the damper, spring, jounce stop, and rebound stop compressions to be defined as functions of jounce or jounce and steering rack travel.



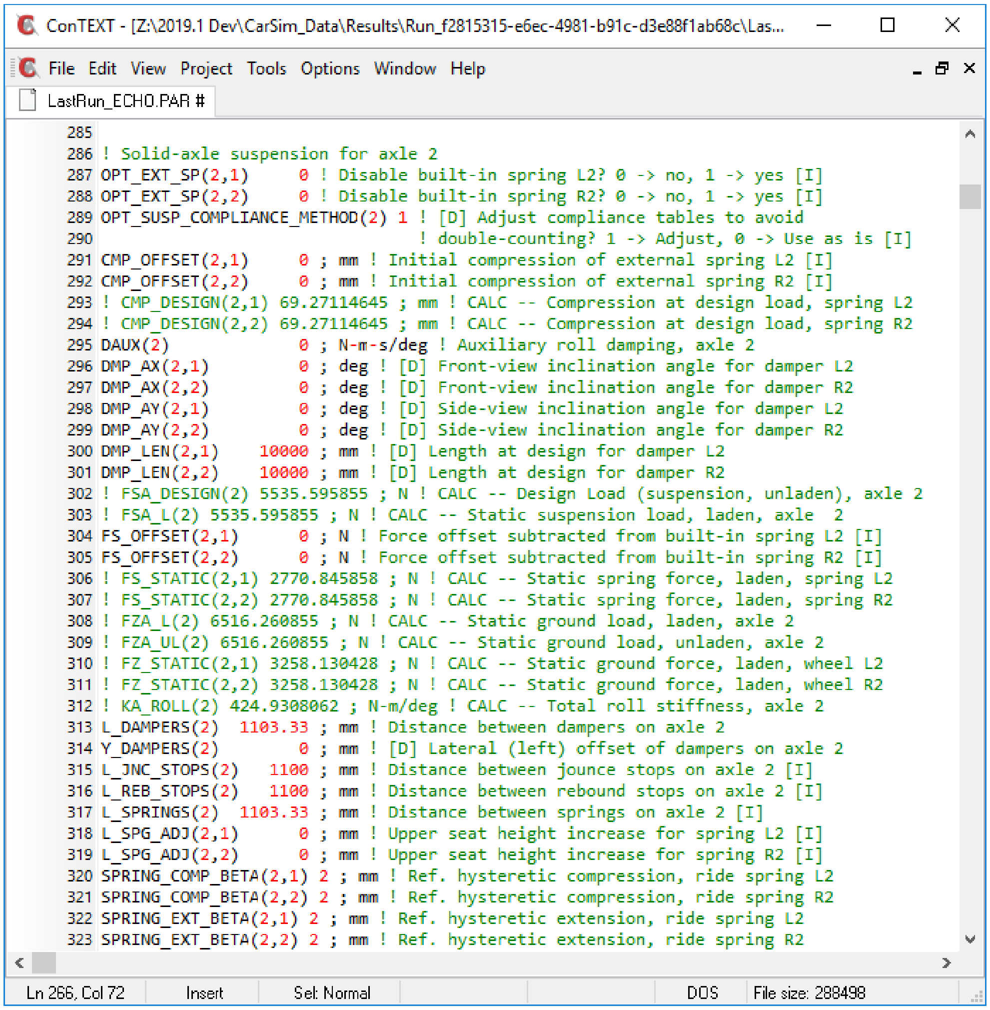

```
!------------------------------------------------------------------------------
! Suspension springs and dampers are specified with the following parameters, along
! with the nonlinear Configurable Functions CMP_DAMP, CMP_JSTOP, CMP_RSTOP,
! CMP_SPR_SEAT, FD, F_JNC_STOP, F_REB_STOP, FS_COMP, FS_EXT, and MX_AUX. All
! suspension models calculate compliance effects using the functions CC_FX, CI_FY,
! CI_MZ, CS_FY, CS_MZ, and CT_FX. Independent suspensions also use the functions
! CD_MY, C_LAT, and C_LONG. Solid-axle suspensions also use the functions
! C_LAT_AXLE and C_LONG_AXLE.

! Generic/independent suspension for axle 1
OPT_EXT_SP(1,1)      0 ! External option for spring L1: 0 -> use built-in spring
                       ! (with or without external model), 1 -> disable built-in
                       ! spring and use an external model [I]
OPT_EXT_SP(1,2)      0 ! Disable built-in spring R1? 0 -> no, 1 -> yes [I]
OPT_SUSP_COMPLIANCE_METHOD(1) 1 ! [D] Subtract offset from each compliance table to
                                ! avoid double-counting kinematical offset? 1 ->
                                ! Yes, subtract the offset, 0 -> No, use compliance
                                ! table as is [I]
CMP_OFFSET(1,1)      0 ; mm ! Initial compression of external spring L1 [I]
CMP_OFFSET(1,2)      0 ; mm ! Initial compression of external spring R1 [I]
! CMP_DESIGN(1,1) 53.4301501 ; mm ! CALC -- Compression at design load, spring L1
! CMP_DESIGN(1,2) 53.4301501 ; mm ! CALC -- Compression at design load, spring R1
DAUX(1)              0 ; N-m-s/deg ! Auxiliary roll damping, axle 1
! FSA_DESIGN(1) 8487.913645 ; N ! CALC -- Design Load (suspension, unladen), axle 1
! FSA_L(1) 8487.913645 ; N ! CALC -- Static suspension load, laden, axle  1
FS_OFFSET(1,1)       0 ; N ! Force offset subtracted from built-in spring L1 [I]
FS_OFFSET(1,2)       0 ; N ! Force offset subtracted from built-in spring R1 [I]
! FS_STATIC(1,1) 6945.919513 ; N ! CALC -- Static spring force, laden, spring L1
! FS_STATIC(1,2) 6945.919513 ; N ! CALC -- Static spring force, laden, spring R1
! FZA_L(1) 9272.445645 ; N ! CALC -- Static ground load, laden, axle 1
! FZA_UL(1) 9272.445645 ; N ! CALC -- Static ground load, unladen, axle 1
! FZ_STATIC(1,1) 4636.222822 ; N ! CALC -- Static ground force, laden, wheel L1
! FZ_STATIC(1,2) 4636.222822 ; N ! CALC -- Static ground force, laden, wheel R1
! KA_ROLL(1) 3347.557624 ; N-m/deg ! CALC -- Total roll stiffness, axle 1
L_SPG_ADJ(1,1)       0 ; mm ! Upper seat height increase for spring L1 to reduce
                       ! spring compression [I]
L_SPG_ADJ(1,2)       0 ; mm ! Upper seat height increase for spring R1 [I]
SPRING_COMP_BETA(1,1) 2 ; mm ! Reference hysteretic compression (1/3 of the
                         ! compression needed to go from the lower force boundary to
                         ! the upper force boundary) for ride spring L1
SPRING_COMP_BETA(1,2) 2 ; mm ! Ref. hysteretic compression, ride spring R1
SPRING_EXT_BETA(1,1) 2 ; mm ! Ref. hysteretic extension, ride spring L1
SPRING_EXT_BETA(1,2) 2 ; mm ! Ref. hysteretic extension, ride spring R1
```

Figure 22. Suspension springs and dampers section of the Echo file (axle 1 is independent).

The Echo files also include a calculated property KA_ROLL — an overall roll stiffness provided by the suspension for an axle. This is based on the spring stiffness properties, the spacing of the springs if it is a solid axle suspension, and an auxiliary roll stiffness provided by the Configurable Function MX_AUX. Details of the auxiliary roll moment are provided in a later section (page 64).

The calculated suspension roll stiffness KA_ROLL is used by the VS Math Model to estimate initial roll angles for vehicles on uneven ground.




```

! Solid-axle suspension for axle 2
OPT_EXT_SP(2,1)      0 ! Disable built-in spring L2? 0 -> no, 1 -> yes [I]
OPT_EXT_SP(2,2)      0 ! Disable built-in spring R2? 0 -> no, 1 -> yes [I]
OPT_SUSP_COMPLIANCE_METHOD(2) 1 ! [D] Adjust compliance tables to avoid
                                ! double-counting? 1 -> Adjust, 0 -> Use as is [I]
CMP_OFFSET(2,1)      0 ; mm ! Initial compression of external spring L2 [I]
CMP_OFFSET(2,2)      0 ; mm ! Initial compression of external spring R2 [I]
! CMP_DESIGN(2,1) 69.27114645 ; mm ! CALC -- Compression at design load, spring L2
! CMP_DESIGN(2,2) 69.27114645 ; mm ! CALC -- Compression at design load, spring R2
DAUX(2)              0 ; N-m-s/deg ! Auxiliary roll damping, axle 2
DMP_AX(2,1)          0 ; deg ! [D] Front-view inclination angle for damper L2
DMP_AX(2,2)          0 ; deg ! [D] Front-view inclination angle for damper R2
DMP_AY(2,1)          0 ; deg ! [D] Side-view inclination angle for damper L2
DMP_AY(2,2)          0 ; deg ! [D] Side-view inclination angle for damper R2
DMP_LEN(2,1)     10000 ; mm ! [D] Length at design for damper L2
DMP_LEN(2,2)     10000 ; mm ! [D] Length at design for damper R2
! FSA_DESIGN(2) 5535.595855 ; N ! CALC -- Design Load (suspension, unladen), axle 2
! FSA_L(2) 5535.595855 ; N ! CALC -- Static suspension load, laden, axle  2
FS_OFFSET(2,1)       0 ; N ! Force offset subtracted from built-in spring L2 [I]
FS_OFFSET(2,2)       0 ; N ! Force offset subtracted from built-in spring R2 [I]
! FS_STATIC(2,1) 2770.845858 ; N ! CALC -- Static spring force, laden, spring L2
! FS_STATIC(2,2) 2770.845858 ; N ! CALC -- Static spring force, laden, spring R2
! FZA_L(2) 6516.260855 ; N ! CALC -- Static ground load, laden, axle 2
! FZA_UL(2) 6516.260855 ; N ! CALC -- Static ground load, unladen, axle 2
! FZ_STATIC(2,1) 3258.130428 ; N ! CALC -- Static ground force, laden, wheel L2
! FZ_STATIC(2,2) 3258.130428 ; N ! CALC -- Static ground force, laden, wheel R2
! KA_ROLL(2) 424.9308062 ; N-m/deg ! CALC -- Total roll stiffness, axle 2
L_DAMPERS(2)   1103.33 ; mm ! Distance between dampers on axle 2
Y_DAMPERS(2)         0 ; mm ! [D] Lateral (left) offset of dampers on axle 2
L_JNC_STOPS(2)    1100 ; mm ! Distance between jounce stops on axle 2 [I]
L_REB_STOPS(2)    1100 ; mm ! Distance between rebound stops on axle 2 [I]
L_SPRINGS(2)   1103.33 ; mm ! Distance between springs on axle 2 [I]
L_SPG_ADJ(2,1)       0 ; mm ! Upper seat height increase for spring L2 [I]
L_SPG_ADJ(2,2)       0 ; mm ! Upper seat height increase for spring R2 [I]
SPRING_COMP_BETA(2,1) 2 ; mm ! Ref. hysteretic compression, ride spring L2
SPRING_COMP_BETA(2,2) 2 ; mm ! Ref. hysteretic compression, ride spring R2
SPRING_EXT_BETA(2,1) 2 ; mm ! Ref. hysteretic extension, ride spring L2
SPRING_EXT_BETA(2,2) 2 ; mm ! Ref. hysteretic extension, ride spring R2
```

Figure 23. Parameters for springs and dampers on a solid axle.

Conventions for Suspension Spring and Damper Screens

The compliance properties of suspensions are assembled on a screen such as the **Suspension: Independent Compliance, Springs, and Dampers** screen (Figure 24).

The main suspension spring can be specified with a dataset from a spring library or with a set of four parameters as shown for the example in Figure 24.

The terms used to describe suspension compliance effects (*toe* or *steer*, *camber* or *inclination*) have been chosen so that for a vehicle with symmetric suspension properties, the numerical signs on the left side and right side coefficients are the same. For example, *toe* for a wheel is defined as positive when the wheel steers inward, toward the vehicle longitudinal axis, while *steer* is defined to be positive when the wheel undergoes a positive rotation about the Z axis (to the left). Fx is defined as a toe effect, because the two wheels on a symmetric suspension will both either steer in or out due to the applied force. Fy is a steer effect, because the applied force will cause wheels of a symmetric suspension to steer both to the left or both to the right.

Camber is defined to be positive when a wheel leans outward at the top. Positive inclination means the wheel leans to the right, a positive rotation about the X axis. For wheels on the right-hand side, camber and inclination are the same for an un-steered wheel; for wheels on the left-hand side, they have opposite signs. Coefficients that involve camber (e.g. Fx) and have positive sign mean that an applied positive force causes the wheel to lean out at the top. Coefficients that involve inclination (e.g., Fy) and have positive sign mean that an applied positive force causes the wheel to lean to the right.

Lateral and longitudinal displacements are measured at the wheel centers. Longitudinal force compliance is measured with equal, parallel longitudinal forces applied at the tire contact patches of both wheels on a suspension. Lateral force compliance is measured with equal but oppositely directed lateral forces applied at the tire contact patches of both wheels on a suspension.

All of the parameters and Configurable Functions that are set on a suspension compliance screen are indexed to the axle number as indicated by the current value of the system parameter `IAXLE`. For models with trailers, they are also indexed to the vehicle unit as indicated by the current value of the system parameter `IUNIT`. Most of the properties can be applied to either the left or right side. The context is specified in the parsfile with the system parameter `ISIDE` (1 = left, 2 = right).

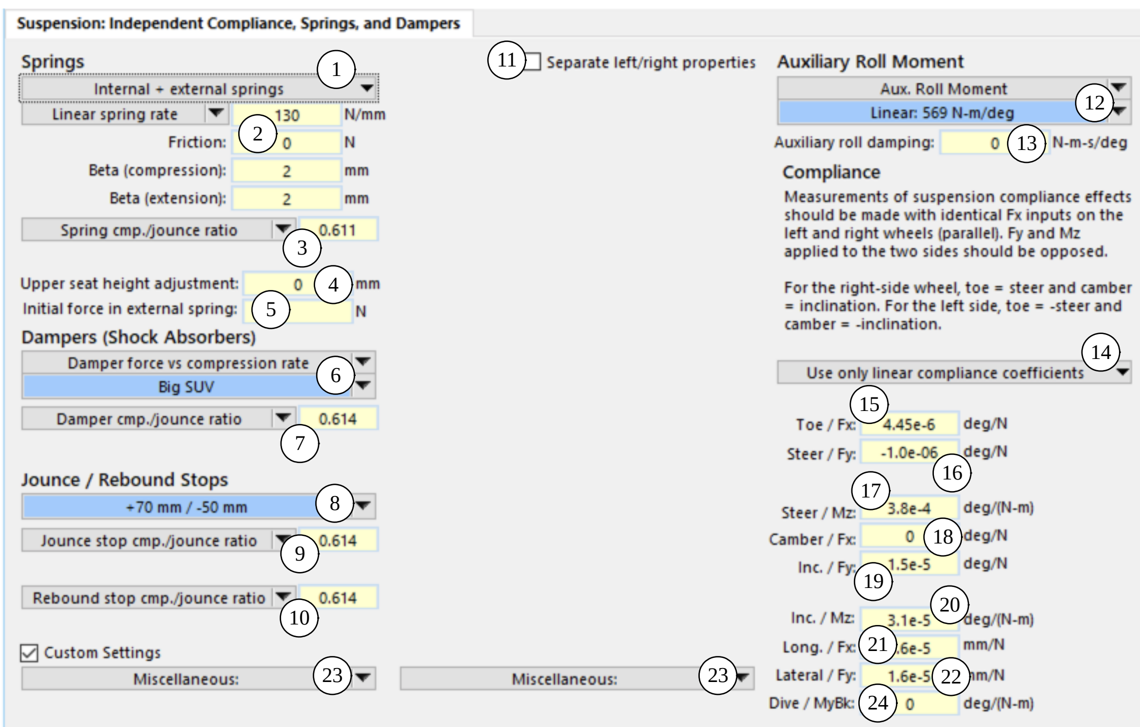


Figure 24. The Suspension: Independent Compliance, Springs, and Dampers screen.

User Settings for All Types of Suspensions

Springs and Dampers

(1) Drop-down control for choosing between three options for spring definitions (Figure 25).

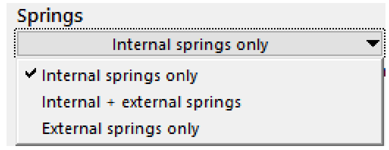


Figure 25. Three options involving external suspension springs.

The first two options specify that the internal hysteretic spring model be used. Selecting either of these causes a drop-down control to appear (2) for specifying data for the internal spring model. The second and third options specify that an external spring model be used. Selecting either of these options causes a data field to be shown for specifying the initial condition for the external spring (5). The type of initial condition depends on whether there is also an internal spring: if there are both internal and externally defined springs, the initial force of each external spring is specified as shown in the figure; if there are only externally defined springs, then the initial spring compression is specified (see the description for the data field (5)). The purpose of the external spring is for use in modeling systems like active suspensions, in which an actuator carries a portion of the suspension load.

Note The indexed math model keyword which determines the presence of an internal spring is `OPT_EXT_SP`. When this is zero, the internal spring is available and may be added with the effects of an external spring using the import variable `IMP_FS_`*i*, where *i* identifies the wheel associated with the spring. When `OPT_EXT_SP` = 1, the internal spring is disabled, and the full spring force should be set with the import `IMP_FS_`*i*. The import variable `IMP_FS_`*i* operates only in "Add" mode, either adding to the internal value (if available), or adding to an internal value of zero (for when the internal spring is disabled).

(2) Settings for specifying data for the internal spring model. A drop-down control specifies whether the function should use a linear spring rate and other parameters as shown in the figure, or a blue link to a **Suspension: Spring** dataset or a **Suspension: Ride Rate (Spring + Tire)** dataset. If the choice is made to use parameters (as shown in Figure 24), the values shown on the screen are used to define two Configurable Functions that specify force vs. deflection when the spring is being compressed and when being extended, as described earlier (Figure 3, page 5). Listing 1 shows how the four values shown on the screen are written to a parsfile for use in the math models:

Listing 1. Portion of parsfile showing spring data.

```
FS_COMP_COEFFICIENT 28
FS_EXT_COEFFICIENT 28
FS_COMP_OFFSET 20
FS_EXT_OFFSET -20
SPRING_COMP_BETA 3
SPRING_EXT_BETA 3
```

1. Spring rate. This value is used to configure both functions (spring compression force and spring extension force) with linear coefficients using the keywords

FS_COMP_COEFFICIENT and FS_EXT_COEFFICIENT. These linear coefficients are listed in the Echo file in the Configurable Function section for the functions FS_COMP and FS_EXT.

2. Friction. This value offsets the two functions, one up and one down using keywords FS_COMP_OFFSET and FS_EXT_OFFSET. These offsets are listed in the Echo file in the Configurable Function section for the functions FS_COMP and FS_EXT.
3. The characteristic deflection in compression is set using the keyword SPRING_COMP_BETA (this is listed with other parameters in the Echo file, e.g., Figure 23).
4. The characteristic deflection in extension is set using the keyword SPRING_EXT_BETA (this is listed with other parameters in the Echo file).

3 Setting to specify spring compression as a Configurable Function of suspension jounce. A drop-down control specifies whether the function should use a linear ratio as shown in the figure (keyword = CMP_SPR_SEAT_COEFFICIENT), or a blue link to a dataset in the **Generic Table** library. If a blue link is used, the root name of the function must be specified in the **Generic Table** dataset: CMP_SPR_SEAT. (CMP_SPR_SEAT_VIR is used instead for the CarSim virtual steering axis suspension.)

If a ratio is used, the value is typically between 0.5 (for some SLA suspensions) and 1.0 (for some MacPherson strut suspensions).

4 Upper spring seat height adjustment (keyword = L_SPG_ADJ). Some vehicles have adjustable spring seats to provide a means of preloading springs. This parameter defines the distance in mm by which the spring seat on the upper side of the spring (sprung mass side) is raised or lowered. A positive value raises the spring seat, extending the spring and reducing its preload.

5 Initial condition of an external spring. When the pull-down control 1 indicates the VS Math Models will be extended through external springs defined in Simulink or external code, then information is needed to allow the VS Math Models to initialize suspension jounce at the start of the simulation run. The required information depends on whether the internal suspension spring is included.

If the option is chosen for **Internal + external springs** (as shown in the figure), then the data field is used to set the initial force produced by the external spring (keyword = FS_OFFSET). This information is combined with the internal spring model to allow the VS Math Model to calculate the initial suspension jounce needed to support the sprung mass.

If the option is chosen for **External springs only**, then the data field is used to set the initial compression of the external spring (keyword = CMP_OFFSET). In this case, the spring compression is used to initialize the position of the suspension and can be exported for use by the external model.

6 Setting to specify damping force from a shock absorber as a Configurable Function of the shock compression rate. A drop-down control specifies whether the function should use a linear coefficient (keyword = FD_COEFFICIENT), or a blue link to a **Shock Absorber** dataset, as shown in the figure.

7 Setting to specify damper (shock absorber) compression as a Configurable Function of suspension jounce. A drop-down control specifies whether the function should use a linear ratio as shown in the figure (keyword = `CMP_DAMP_COEFFICIENT`), or a blue link to a dataset in the **Generic Table** library. If a blue link is used, the root name of the function must be specified in the **Generic Table** dataset: `CMP_DAMP`. (`CMP_DAMP_VIR` is used instead for the CarSim virtual steering axis suspension.)

If a ratio is used, the value is typically between 0.5 (for some SLA suspensions) and 1.0 (for some MacPherson strut suspensions).

8 Link to a **Suspension: Jounce and Rebound Stops** dataset. Jounce and rebound forces operate directly on each axle or suspension, as described in an earlier subsection (page 10). In the tables, "Jounce stop compression" and "Rebound stop compression" refer to displacements at the location of the stop, *not* wheel center displacement. The most common example is of jounce and rebound stops incorporated into a damper that has a motion ratio not equal to 1. The compression values in the table would reflect actual damper displacements.

9 Setting to specify compression of the jounce stop bumper as a Configurable Function of suspension jounce. A drop-down control specifies whether the function should use a linear ratio as shown in the figure (keyword = `CMP_JSTOP_COEFFICIENT`), or a blue link to a dataset in the **Generic Table** library. If a blue link is used, the root name of the function must be specified in the **Generic Table** dataset: `CMP_JSTOP`. (`CMP_JSTOP_VIR` is used instead for the CarSim virtual steering axis suspension.)

10 Setting to specify compression of the rebound stop bumper as a Configurable Function of suspension jounce. A drop-down control specifies whether the function should use a linear ratio as shown in the figure (keyword = `CMP_RSTOP_COEFFICIENT`), or a blue link to a dataset in the **Generic Table** library. If a blue link is used, the root name of the function must be specified in the **Generic Table** dataset: `CMP_RSTOP`. (`CMP_RSTOP_VIR` is used instead for the CarSim virtual steering axis suspension.)

11 Checkbox to specify whether different properties should be specified for the left and right sides of the suspension. When unchecked (as in the figure), the specified settings described above (1 - 10) are written twice in the dataset for use on both sides. When checked, separate settings are shown in the screen for the left and right sides.

12 Link to a **Suspension: Auxiliary Roll Moment** dataset or a **Suspension: Measured Total Roll Stiffness** dataset. The linked dataset accounts for the difference between the overall roll stiffness and the stiffness provided by the springs alone.

13 Auxiliary roll damping (keyword = `DAUX`). This parameter accounts for the difference between the overall roll damping and the roll damping provided by the shock absorbers alone.

Compliance Values

The CarSim and TruckSim suspension models include over twenty compliance properties. A few are specific to the type of suspension. For example, the dive compliance 24 is used only for independent and virtual steering axis suspensions. However, all types of suspensions share most of the compliance properties. These compliances are represented in the math models with Configurable Functions, in which a deflection is a potentially nonlinear function of applied forces or moments, as described earlier. However, most users find that linear coefficients are adequate for representing the compliance effects. If linear coefficients are used, the values can all be set on this screen.

14 Drop-down control to choose source of data for compliances. This control has two options: only linear coefficients (as shown in the figure), or using some nonlinear tables. If the option to use some nonlinear tables is chosen, a blue link is shown for linking to a dataset in the **Suspension: Compliance (Nonlinear)** library (Figure 40, page 67) and the yellow data fields shown in the figure are hidden.

15 Coefficient for change in toe per change of tire longitudinal force (keyword = `CT_FX_COEFFICIENT`). A forward tractive force tends to bend a suspension forward, steering the wheel inward (positive toe). Therefore, this parameter is likely to have a small but positive value.

16 Coefficient for change of steer angle per change of tire lateral force (keyword = `CS_FY_COEFFICIENT`). For wheels that can be steered, the steer axis, virtual or otherwise, is usually inclined to intersect the ground in front of the center of tire contact. Thus, a positive lateral force (to the left), acting behind the steer axis usually causes some steer to the right (negative). Therefore, this coefficient is likely to have a small negative value for a steered wheel. For un-steered rear wheels, it should have a value close to zero.

17 Coefficient for change of steer angle per change of tire aligning torque (keyword = `CS_MZ_COEFFICIENT`). The suspension elements usually deflect when a steering torque is applied to the wheel. Because the steer and moment have the same sign convention, the compliance coefficient is nearly always positive.

Aligning torque affects steering due to both compliance in the suspension and compliance in the steering system. The effect of this suspension coefficient leads to a steering angle that is added to the steering due to other factors, including the steering system compliance as described in the **Steering** screen.

18 Coefficient for change in camber per change of tire longitudinal force (keyword = `CC_FX_COEFFICIENT`). This parameter is likely to have a value close to zero.

19 Coefficient for change of inclination angle per change of tire lateral force (keyword = `CI_FY_COEFFICIENT`). Positive lateral force (to the left) tends to bend the suspension and let the wheel incline to the right (positive inclination). Therefore, this parameter is expected to have a small positive value.

20 Coefficient for change of inclination angle per change of tire aligning torque (keyword = `CI_MZ_COEFFICIENT`). This parameter is likely to have a value close to zero.

21 Coefficient for longitudinal displacement of the wheel center per change of tire longitudinal force (keyword = `C_LONG_COEFFICIENT`). Displacement usually occurs in the same direction as the force, and displacements and forces have the same convention, so this parameter usually has a small positive value. Longitudinal force compliance is measured with equal, parallel longitudinal forces applied at the tire contact patches of both wheels on a suspension.

22 Coefficient for lateral displacement of the wheel center per change of tire lateral force (keyword = `C_LAT_COEFFICIENT`). Displacement usually occurs in the same direction as the force, and displacements and forces have the same convention, so this parameter almost always has a small positive value. Lateral force compliance is measured with equal but oppositely directed lateral forces applied at the tire contact patches of both wheels on a suspension.

Custom Settings

23 Options for custom settings. Use the checkbox to display two miscellaneous links. The links have no predefined purpose but can be linked to datasets that specify parameters for extensions to the model or information for the animator. The links are not displayed unless the box is checked.

Suspension: Independent Compliance, Springs, and Dampers

The screen for specifying compliance properties for independent suspensions was shown in Figure 24 (page 42). All of the items described in the previous subsection are common to this screen and others for solid axle and twist-beam suspensions. The screen for independent suspensions contains one type of compliance that is unique to this suspension model: dive compliance.

24 Coefficient for dive (pitch) displacement of the un-steered part of the suspension due to braking torque (keyword = `CD_MY_COEFFICIENT`). Displacement usually occurs in the same direction as the moment. For a steered wheel, this dive also applies to caster.

This parameter usually has a small positive value.

Suspension: Solid Axle Compliance, Springs, and Dampers

The screen for specifying compliance properties for solid axle suspensions (Figure 26) is almost the same as the example discussed above. The differences are:

1. There is no dive compliance.
2. The lateral and longitudinal translational compliances (21 and 22) apply for the whole suspension, not just one side. (This is seen when the box is checked to show separated properties on the left and right sides 11.)
3. Additional data fields are shown to specify the spacing of the springs, dampers, and stops on the axle (25 - 28).
4. If a dataset is linked to an axle that is not steered, the compliances for `Toe/Fx` 15, `Steer/Fy` 16, `Steer/Mz` 17, `Camber/Fx` 18, `Inc./Fy` 19, and `Inc/Mz` 20 are ignored, because an unsteered axle lacks the multibody degrees of freedom to support them. If these are necessary on an unsteered axle, setting the keyword `OPT_STEER_EXT(id)` to zero causes the more detailed suspension module to be used, and the compliances are supported. For example, on a vehicle without a trailer "`OPT_STEER_EXT(2)  0`" sets the second axle to the detailed module, "`OPT_STEER_EXT(2,1)  0`", sets the first axle on the second vehicle unit. In CarSim, you can also select the "Solid axle (with wheel compliance)" type from the vehicle assembly

screen. Note that any axle specified as a steer axle always receives the detailed module by default.

Solid Axle Spring and Damper Geometry

The solid axle model has two conceptual approaches to modeling the ride springs and ride dampers: *system-level* and *geometric*. The system level approach, introduced in version 2021.1, uses 2D table types for the spring compression (`CMP_SPR_SEAT`) and damper compression (`CMP_DAMP`). These are the same math model keywords as used by the generic/independent suspension type; consequently, the independent variables when a 2D table type is specified are jounce of the wheel and the jounce of the wheel on the other side. This implies the two solid-axle DOFs of axle jounce and axle roll.

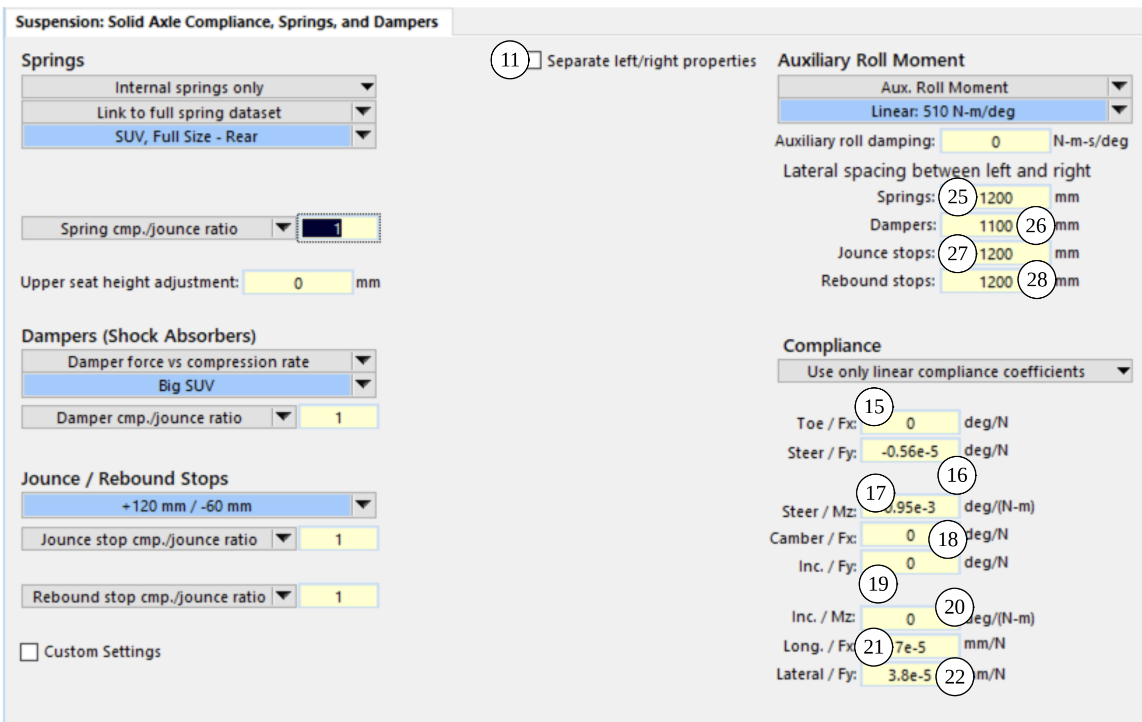


Figure 26. The Suspension: Solid Axle Compliance, Springs, and Dampers screen.

The geometric approach uses 1D table types (or even just coefficients) for the spring compression and damper compression keywords of `CMP_SPR_SEAT` and `CMP_DAMP`. When using the 1D table type for these with the solid axle, it is necessary to enter additional geometric parameters associated with the springs and dampers. Referring to the **Solid Axle Compliance** screen of Figure 26:

25 Spring spacing (keyword = `L_SPRINGS`). Lateral distance between the centerlines of left and right springs. Advanced users may offset the two springs laterally to provide an asymmetric effect using the keyword `Y_SPRINGS`, which must be entered in a miscellaneous yellow field.

26 Shock absorber spacing (keyword = `L_DAMPERS`). Lateral distance between the centerlines of the left and right shocks. Advanced users may offset the two dampers laterally to provide an asymmetric effect using the keyword `Y_DAMPERS`, which must be entered in a miscellaneous yellow field.

Note If you are using the 2D table type for the spring or damper compression, you should set the spacing to the nominal lateral distance between the wheel centers. In other words, the spacing should match the track width value entered on the **Solid Axle Kinematics** screen, math model keyword `L_TRACK`. You can type the `L_TRACK` keyword directly into the yellow field on the compliance screen; the math model will then use the value that was entered on the kinematics screen.

For advanced users of the geometric approach, the solid axle damper specifications support a more detailed mode with inclined mounting.

- Solid axle roll damping can be influenced significantly if the dampers are mounted at an angle, especially on suspensions that exhibit substantial lateral movement with roll. Examples include some four-link solid axles and the NASCAR-type truck arm suspension. Inclining the dampers inward at the top moves the line of action of the damper force toward the roll center of the axle, reducing the roll moment contribution of the damper force.
- To specify inclined dampers for a solid axle, you must supply values for three parameters for each shock by typing them into a miscellaneous yellow field or reading an external Parsfile. For example, use the **Custom Settings** checkbox on the **Solid Axle Compliance** screen to link to another dataset with the additional damper settings.
- The parameters are `DMP_AX`, `DMP_AY`, and `DMP_LEN`. `DMP_AX` is the angle between the damper and vertical when viewed from the front of the vehicle looking back and is positive when the damper is inclined inward at the top. `DMP_AY` is the angle between the damper and vertical when viewed from the side and is positive when the damper is inclined forward at the top. `DMP_LEN` is the length of the damper when the axle is at its design position. The top of the damper is always assumed to be mounted to the chassis, and the bottom to the axle.
- The math model defaults are zero for both angles and for the length. Using a damper length of zero disables the inclined damper effect entirely; setting nonzero angles in this case will have no effect. Each damper on an axle (left/right) must have the same `DMP_LEN` setting (zero/greater-than-zero) for the model to work correctly.

Note Versions of CarSim and TruckSim from 8.1 to 2020.0 used a default value of 10 meters for the `DMP_LEN` parameter. This long default length allowed the inclined damper model to approximate the pre-8.1 behavior. Beginning in CarSim and TruckSim versions 2020.1, the pre-8.1 non-inclined damper model has been added back as another damper option. The non-inclined damper is activated by specifying `DMP_LEN` equal to zero, while the inclined damper is activated by specifying `DMP_LEN` greater than zero. Consequently, setting `DMP_LEN` to 10 meters restores the calculation method used from versions 8.1 to 2020.0.

Similarly, *for advanced users of the geometric approach*, the solid axle spring specifications support a more detailed mode with inclined mounting (introduced in version 2020.1):

- To specify inclined springs for a solid axle, supply values for three parameters for each spring unit by typing them into a yellow field or by reading them from an external parsfile. For example, use the **Custom Settings** checkbox on the **Solid Axle Compliance** screen to link to a dataset with the additional spring and/or damper settings.
- The spring inclination parameters are
  - SPG_AX, the angle between the spring and vertical when viewed from the front of the vehicle looking back, positive when the spring is inclined inward at the top
  - SPG_AY, the angle between the spring and vertical when viewed from the side, positive when the spring is inclined forward at the top
  - SPG_LEN, the length of the spring when the axle is at its design position.
- As with the dampers, the top of the spring is always assumed to be mounted to the chassis, and the bottom to the axle. Additionally, setting SPG_LEN to zero uses the non-inclined spring model, and specifying SPG_LEN greater than zero activates the inclined spring model. Each spring on an axle (left/right) must have the same SPG_LEN setting (zero/greater-than-zero) for the model to work correctly.

> **Note** When using the inclined dampers or springs, carefully consider the installation ratio parameters CMP_SPR_SEAT_COEFFICIENT 3 and CMP_DAMP_COEFFICIENT 7, referring to Figure 24 (page 42). These establish the unit of spring or damper compression per unit of wheel jounce for the spring and damper, respectively, meaning it is possible to double-count the installation ratio effects if using the inclined spring and/or damper options. (The inclined spring/damper model calculates a compression value based on the actual geometry provided.) In other words, the compression coefficient should be set to one if the detailed length/angle geometry is active.
>
> Additionally, the inclined spring option, enabled by setting SPG_LEN to a value greater than zero, is not recommended for tandem or tridem suspensions, as the load sharing behavior established by R_TANDEM assumes that the springs are mounted vertically.

The various spring and damper concepts and options are summarized in Table 2, based on the modeling approach chosen. Example keywords are for a damper, but analogous for a spring.

Table 2. The spring or damper geometry settings for the solid axle type,

	System level	**Geometric: simple**	**Geometric: detailed**
Compression *E.g.,* CMP_DMP	2D (wheel jounce, wheel jounce other side)	1D (wheel jounce)	1D (wheel jounce), but set coefficient equal to one
Spacing *E.g.,* L_DAMPERS	= track width (L_TRACK)	Actual	Actual
Lateral offset *E.g.,* Y_DAMPERS	= 0	Actual	Actual
Length *E.g.,* DMP_LEN	= 0 (to disable detailed geometry)	= 0 (to disable detailed geometry)	Actual (nonzero to enable detailed geometry)
Front-view inclination *E.g.,* DMP_AX	= 0	= 0	Actual
Side-view inclination *E.g.,* DMP_AY	= 0	= 0	Actual

Solid Axle Jounce and Rebound Stop Geometry

The **Solid Axle Compliance** screen, like that for independent suspension model, also lets you input the jounce and rebound stop compressions. The difference is that the solid axle model needs spacing parameters depending on how the data is entered. This is because the jounce and rebound stop compressions (CMP_JSTOP and CMP_RSTOP) support 1D (jounce) or 2D (jounce, jounce other side) table types, like the springs and dampers described in the previous section. The 1D table type requires the spacing parameters for the solid axle. Referring to the **Solid Axle Compliance** screen of Figure 26 (page 48):

(27) Jounce bump stop spacing (keyword = L_JNC_STOPS). Lateral distance between the centerlines of left and right jounce bump stops.

(28) Rebound bump stop spacing (keyword = L_REB_STOPS). Lateral distance between the centerlines of left and right rebound bump stops.

> **Note** If you are using the 2D table type for the jounce stop or rebound stop compression, you should set the corresponding spacing to the nominal lateral distance between the wheel centers. In other words, the spacing should match the track width value entered on the **Solid Axle Kinematics** screen, math model keyword L_TRACK. You can type the L_TRACK keyword directly into the yellow field on the compliance screen; the math model will then use the value that was entered on the kinematics screen.

Suspension: Virtual Steering Axis Compliance, Springs, and Dampers

The virtual steering axis suspension is not available in TruckSim.

The Suspension: Virtual Steering Axis Compliance, Springs, and Dampers screen is visually identical to the Suspension: Independent Compliance, Springs, and Dampers screen, Figure 24. The only notable difference is that this screen writes CMP_DAMP_VIR, CMP_SPR_SEAT_VIR, CMP_JSTOP_VIR, and CMP_RSTOP_VIR Configurable Function root keywords instead of CMP_DAMP, CMP_SPR_SEAT, CMP_JSTOP, and CMP_RSTOP, respectively, as discussed in the section Screens for Suspension Compliance, Springs, and Dampers (page 39). This is because these properties may be entered as 2D functions of both jounce and steering rack travel.

Suspension: Twist Beam Compliance, Springs, and Dampers (2016)

This twist beam suspension screen is not available in TruckSim.

> **Note** As a reminder, there are three twist beam approaches: generic/independent suspension with 2D kinematics (2018), generic/independent with twist beam screens (2016), and legacy twist beam. This section covers the 2016 model.

Twist beams present unique challenges in multibody models. All moving bodies are assumed to be rigid, connected by joints that constrain one or more degrees of freedom, and acted on by various forces. The joints and constraints describe the kinematic behavior of the system. Bodies move within their constrained spaces as they are acted upon by forces.

A twist beam suspension has only one kinematic degree of freedom: it can rotate about the bushings that attach it to the chassis. Remember, kinematics describes *motion in the absence of force*. This single kinematic degree of freedom allows pitch motion, but would not, for example, allow roll. All of the other motions that describe the orientation of the wheels (steer, camber, lateral displacement, longitudinal displacement) result from structural flexibility of the beam components.

In CarSim, specifications of models are based on measurable parameters whenever possible. Many users do not ordinarily have access to finite element models of components. CarSim does, however, have a means to account for the effects of structural flexibility in suspensions: its quasi-static compliance models.

The individual compliance effects in CarSim are supported by Configurable Functions, meaning they can be extensively modified using VS Commands.

We observe that the structural deformation in twist beams are linear, or nearly so (parts are not stressed to yield, deflections are not large). The responses of a linear system can be decomposed into the responses to individual inputs. The responses (deformations) are calculated separately, then superposed to obtain the complete orientation of the wheel.

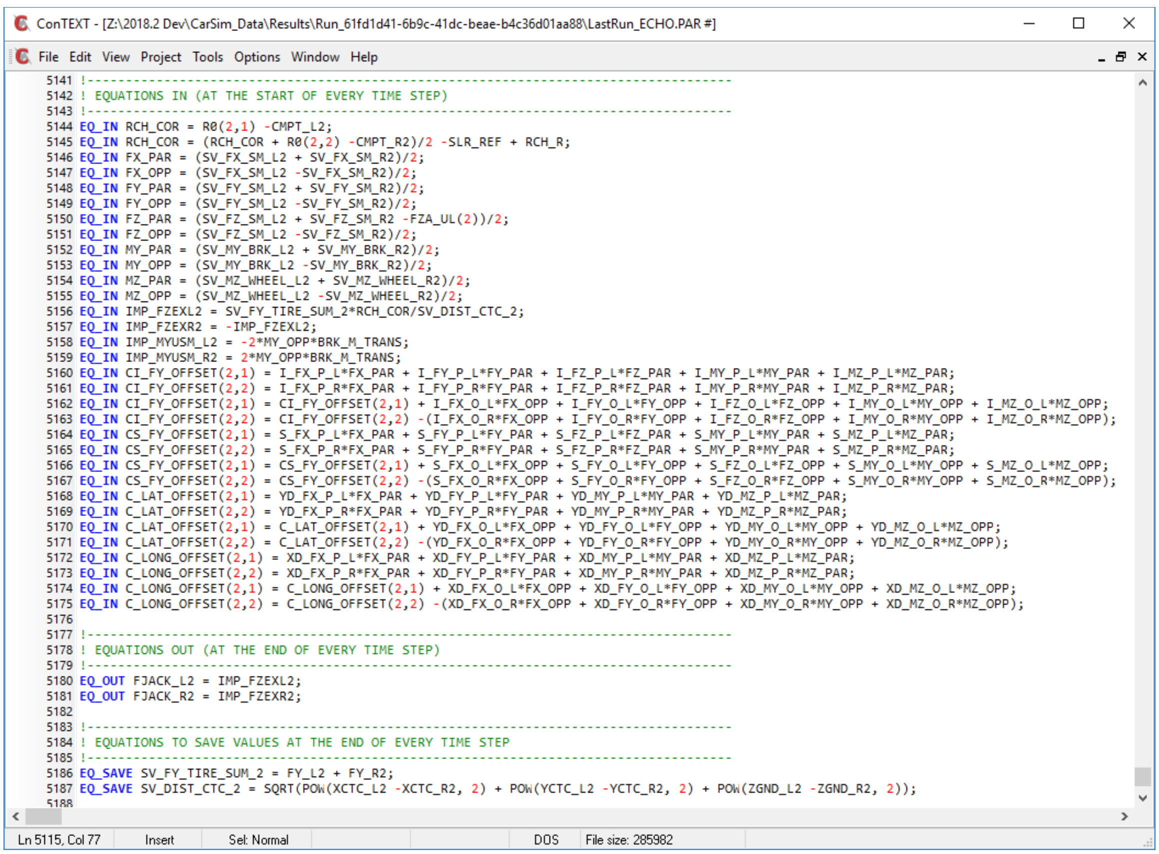
We also observe that the twist beam has two different primary modes of deflection for each type of load (steer, inclination, lateral, longitudinal), depending on the direction of the applied loads (parallel or opposed). Visualize the beam in response to opposed lateral forces, and compare the response to parallel forces. For the opposed case, think of a simply supported beam with oppositely directed bending moments at the ends. For the parallel case, it resembles a beam subjected to opposing transverse shear loads at the ends. The beam exhibits different amounts of stiffness for the two modes.

Fortunately, in a linear model we can decompose the applied loads into a parallel part and an opposed part. The response to each part of the total is calculated and then added together. This is done for each component load (Fx, Fy, Fz, Mx, My, Mz). The sum of all the deflections is a quasi-static linear model of the beam deflection. For the model, the Mx moments are ignored, as they are assumed to be small. For the lateral and longitudinal loads, the Fz force is ignored as well. For a model, compliance values are set in the various Twist Beam Suspension Compliance screens.

This implementation uses VS Commands to make use of offsets to some of the Configurable Functions for compliance that are part of the independent suspension model. For example, Figure 27 shows the end of an Echo file with equations that add inclination (`CI_FY_OFFSET`), steer (`CS_FY_OFFSET`), lateral displacement (`C_LAT_OFFSET`), and longitudinal offset (`C_LONG_OFFSET`) for the two rear wheels.

These equations make use of existing import variables for forces and moments acting on the rear wheels, along with 75 parameters automatically added to the model via the `DEFINE_PARAMETER` command (Figure 28).

The screen for specifying compliance properties for twist beam suspensions has most features of the independent suspension screen, with a few differences. Instead of setting linear or non-linear (tables) coefficients for a limited set of compliances, linear coefficients are set for a more comprehensive set of compliances.




```
!----------------------------------------------------------------------------------
! EQUATIONS IN (AT THE START OF EVERY TIME STEP)
!----------------------------------------------------------------------------------
EQ_IN RCH_COR = R0(2,1) -CMPT_L2;
EQ_IN RCH_COR = (RCH_COR + R0(2,2) -CMPT_R2)/2 -SLR_REF + RCH_R;
EQ_IN FX_PAR = (SV_FX_SM_L2 + SV_FX_SM_R2)/2;
EQ_IN FX_OPP = (SV_FX_SM_L2 -SV_FX_SM_R2)/2;
EQ_IN FY_PAR = (SV_FY_SM_L2 + SV_FY_SM_R2)/2;
EQ_IN FY_OPP = (SV_FY_SM_L2 -SV_FY_SM_R2)/2;
EQ_IN FZ_PAR = (SV_FZ_SM_L2 + SV_FZ_SM_R2 -FZA_UL(2))/2;
EQ_IN FZ_OPP = (SV_FZ_SM_L2 -SV_FZ_SM_R2)/2;
EQ_IN MY_PAR = (SV_MY_BRK_L2 + SV_MY_BRK_R2)/2;
EQ_IN MY_OPP = (SV_MY_BRK_L2 -SV_MY_BRK_R2)/2;
EQ_IN MZ_PAR = (SV_MZ_WHEEL_L2 + SV_MZ_WHEEL_R2)/2;
EQ_IN MZ_OPP = (SV_MZ_WHEEL_L2 -SV_MZ_WHEEL_R2)/2;
EQ_IN IMP_FZEXL2 = SV_FY_TIRE_SUM_2*RCH_COR/SV_DIST_CTC_2;
EQ_IN IMP_FZEXR2 = -IMP_FZEXL2;
EQ_IN IMP_MYUSM_L2 = -2*MY_OPP*BRK_M_TRANS;
EQ_IN IMP_MYUSM_R2 = 2*MY_OPP*BRK_M_TRANS;
EQ_IN CI_FY_OFFSET(2,1) = I_FX_P_L*FX_PAR + I_FY_P_L*FY_PAR + I_FZ_P_L*FZ_PAR + I_MY_P_L*MY_PAR + I_MZ_P_L*MZ_PAR;
EQ_IN CI_FY_OFFSET(2,2) = I_FX_P_R*FX_PAR + I_FY_P_R*FY_PAR + I_FZ_P_R*FZ_PAR + I_MY_P_R*MY_PAR + I_MZ_P_R*MZ_PAR;
EQ_IN CI_FY_OFFSET(2,1) = CI_FY_OFFSET(2,1) + I_FX_O_L*FX_OPP + I_FY_O_L*FY_OPP + I_FZ_O_L*FZ_OPP + I_MY_O_L*MY_OPP + I_MZ_O_L*MZ_OPP;
EQ_IN CI_FY_OFFSET(2,2) = CI_FY_OFFSET(2,2) -(I_FX_O_R*FX_OPP + I_FY_O_R*FY_OPP + I_FZ_O_R*FZ_OPP + I_MY_O_R*MY_OPP + I_MZ_O_R*MZ_OPP);
EQ_IN CS_FY_OFFSET(2,1) = S_FX_P_L*FX_PAR + S_FY_P_L*FY_PAR + S_FZ_P_L*FZ_PAR + S_MY_P_L*MY_PAR + S_MZ_P_L*MZ_PAR;
EQ_IN CS_FY_OFFSET(2,2) = S_FX_P_R*FX_PAR + S_FY_P_R*FY_PAR + S_FZ_P_R*FZ_PAR + S_MY_P_R*MY_PAR + S_MZ_P_R*MZ_PAR;
EQ_IN CS_FY_OFFSET(2,1) = CS_FY_OFFSET(2,1) + S_FX_O_L*FX_OPP + S_FY_O_L*FY_OPP + S_FZ_O_L*FZ_OPP + S_MY_O_L*MY_OPP + S_MZ_O_L*MZ_OPP;
EQ_IN CS_FY_OFFSET(2,2) = CS_FY_OFFSET(2,2) -(S_FX_O_R*FX_OPP + S_FY_O_R*FY_OPP + S_FZ_O_R*FZ_OPP + S_MY_O_R*MY_OPP + S_MZ_O_R*MZ_OPP);
EQ_IN C_LAT_OFFSET(2,1) = YD_FX_P_L*FX_PAR + YD_FY_P_L*FY_PAR + YD_MY_P_L*MY_PAR + YD_MZ_P_L*MZ_PAR;
EQ_IN C_LAT_OFFSET(2,2) = YD_FX_P_R*FX_PAR + YD_FY_P_R*FY_PAR + YD_MY_P_R*MY_PAR + YD_MZ_P_R*MZ_PAR;
EQ_IN C_LAT_OFFSET(2,1) = C_LAT_OFFSET(2,1) + YD_FX_O_L*FX_OPP + YD_FY_O_L*FY_OPP + YD_MY_O_L*MY_OPP + YD_MZ_O_L*MZ_OPP;
EQ_IN C_LAT_OFFSET(2,2) = C_LAT_OFFSET(2,2) -(YD_FX_O_R*FX_OPP + YD_FY_O_R*FY_OPP + YD_MY_O_R*MY_OPP + YD_MZ_O_R*MZ_OPP);
EQ_IN C_LONG_OFFSET(2,1) = XD_FX_P_L*FX_PAR + XD_FY_P_L*FY_PAR + XD_MY_P_L*MY_PAR + XD_MZ_P_L*MZ_PAR;
EQ_IN C_LONG_OFFSET(2,2) = XD_FX_P_R*FX_PAR + XD_FY_P_R*FY_PAR + XD_MY_P_R*MY_PAR + XD_MZ_P_R*MZ_PAR;
EQ_IN C_LONG_OFFSET(2,1) = C_LONG_OFFSET(2,1) + XD_FX_O_L*FX_OPP + XD_FY_O_L*FY_OPP + XD_MY_O_L*MY_OPP + XD_MZ_O_L*MZ_OPP;
EQ_IN C_LONG_OFFSET(2,2) = C_LONG_OFFSET(2,2) -(XD_FX_O_R*FX_OPP + XD_FY_O_R*FY_OPP + XD_MY_O_R*MY_OPP + XD_MZ_O_R*MZ_OPP);

!----------------------------------------------------------------------------------
! EQUATIONS OUT (AT THE END OF EVERY TIME STEP)
!----------------------------------------------------------------------------------
EQ_OUT FJACK_L2 = IMP_FZEXL2;
EQ_OUT FJACK_R2 = IMP_FZEXR2;

!----------------------------------------------------------------------------------
! EQUATIONS TO SAVE VALUES AT THE END OF EVERY TIME STEP
!----------------------------------------------------------------------------------
EQ_SAVE SV_FY_TIRE_SUM_2 = FY_L2 + FY_R2;
EQ_SAVE SV_DIST_CTC_2 = SQRT(POW(XCTC_L2 -XCTC_R2, 2) + POW(YCTC_L2 -YCTC_R2, 2) + POW(ZGND_L2 -ZGND_R2, 2));
```

Figure 27. VS Commands to add inclination, steer, lateral displacement, and longitudinal displacement for the wheels in a twist beam suspension.

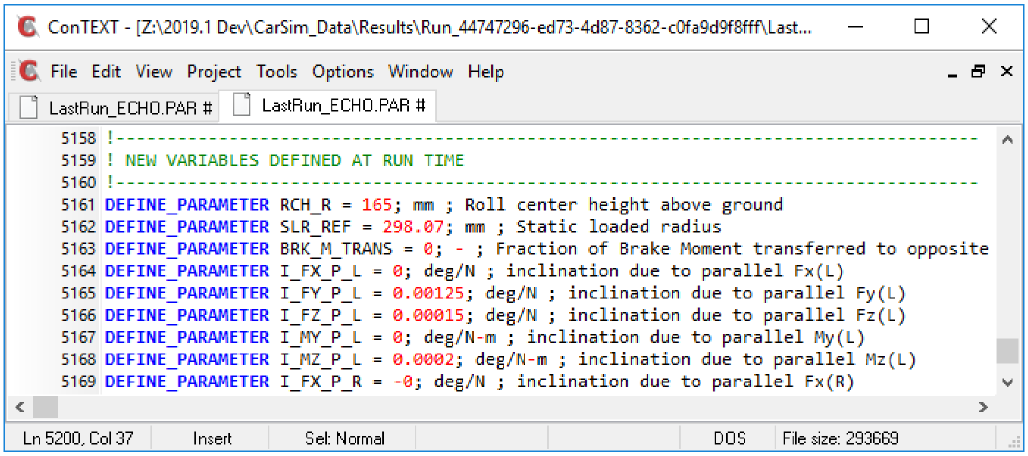


Figure 28. The 2016 twist-beam model applies 75 DEFINE_PARAMETER commands.

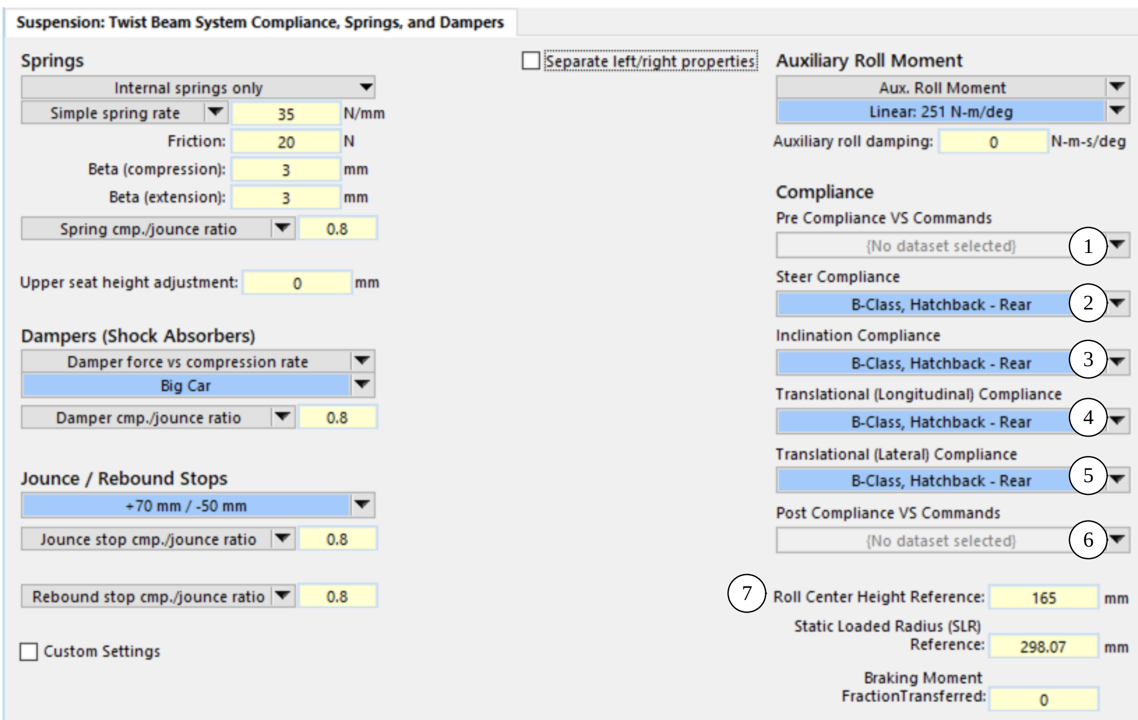


Figure 29. The Suspension: Twist Beam Compliance, Springs, and Dampers screen.

The Twist Beam model compliance screen (Figure 29) has yellow fields to set some of the key values of the Twist Beam model, as well as blue links to allow the setting of several other compliances needed for the model. Multiple blue links are used because many compliance settings are needed to define the Twist Beam model.

1. A link to a screen of general VS Commands (1) is provided to allow an advanced user to apply VS Commands before the special compliance links are read.

2. Four separate links for compliances related to Steer (2), Inclination (3), Longitudinal Translation (4), and Lateral Translation (5) are provided with this screen. These compliance screens are similar, and we will describe one in greater detail.

3. Another link to a screen of general VS Commands (6) is provided to allow advanced users to apply VS Commands after the special compliance links are read.

4. In addition to the blue fields, three yellow fields are used to provide key parameters for the model (7). These are the Roll Center Height (expressed in mm), Static Loaded Radius (also expressed in mm) and Braking Moment Fraction Transferred (expressed as a fraction for each side). The yellow fields are written out before any of the blue compliance links are written.

   The Roll Center Height and Static Loaded Radius are not properties of a suspension alone, as they depend on information about tires and load. The reference values here represent the properties of the vehicle *as tested*. They are used to calculate effects correctly on vehicles with different tires and loading.

   The Braking Moment Fraction Transferred parameter describes the fraction of a braking moment applied to one wheel that is reacted at the opposite side of the suspension, due to the structural flexibility of the beam.

The Twist Beam Compliance Steer screen is shown in Figure 30.

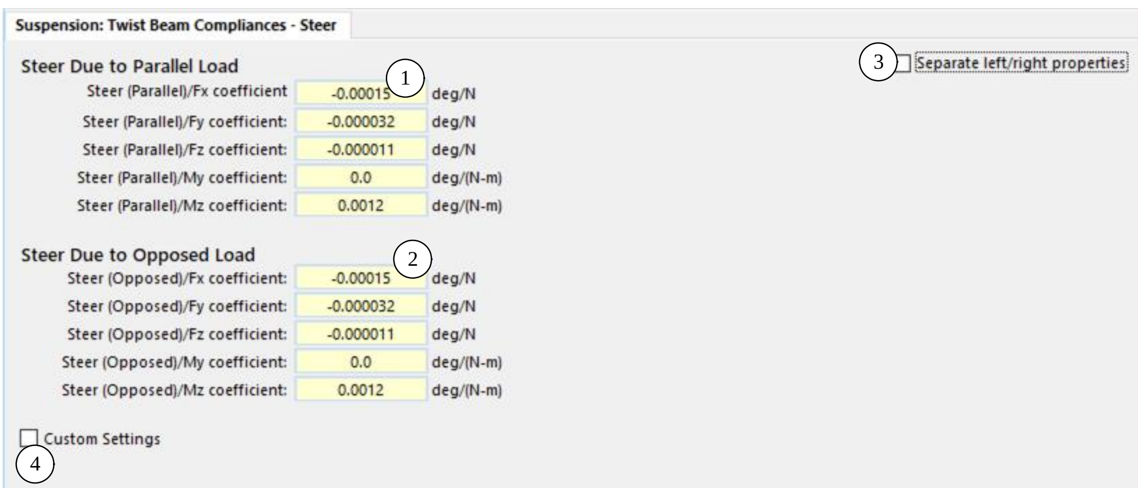


Figure 30. Twist Beam Compliance (Steer) screen.

A set of five parameters represent the steer compliances for parallel loads (1) for a given side. Similarly, a set of five parameters are used to represent the steer compliances for opposed loads (2). If different (asymmetric) compliances are to be set for the other side, the left/right check box (3) allows for values for the compliances to be set for the left and right sides separately. Finally, a Custom Settings check box (4) reveals a miscellaneous yellow field and a miscellaneous link. These allow the user to add VS Commands after the other fields of this screen are written.

The other compliance screens (Inclination, Longitudinal, Lateral) are organized like the Steer screen and operate in a similar manner. Accounting for left and right sides for each grouping, a total of 75 parameters are used to define wheel movements in this Twist Beam model.

Suspension: Twist Beam Compliance, Springs, and Dampers (Legacy)

This twist beam suspension is not available in TruckSim.

The screen for specifying compliance properties for legacy twist beam suspensions (Figure 31) has most features of the **Solid Axle Compliance** screen, with a few differences:

1. The lateral and longitudinal translational compliances (21 and 22) each apply for just one side, as is the case for the independent suspension.
2. Additional data fields are shown to specify compliances due to a change in vertical tire force (29 and 30).
3. Fields are shown for specifying the lateral spacing between left and right components, as done with solid-axle suspensions (25 - 28). However, they sometime serve to compensate for structure flexibility and are not always set to the physical dimension. They are discussed below.

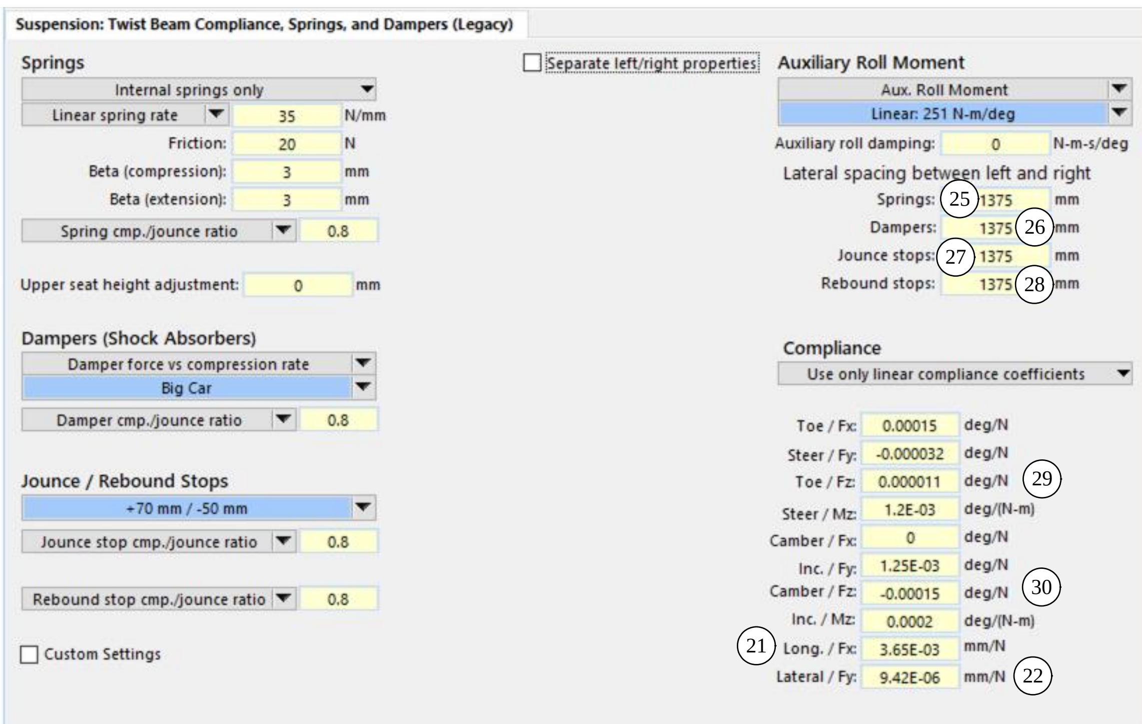


Figure 31. The Suspension: Twist Beam Compliance, Springs, and Dampers (Legacy) screen.

User Controls Unique for This Screen

29 Coefficient for change of toe angle per change of tire vertical force (keyword = `CT_FZ_COEFFICIENT`). Sometimes called ride steer or ride toe, this parameter is used to describe toe that results from structural deflection of the twist beam assembly under vertical load. This parameter is likely to have a value close to zero.

30 Coefficient for change in camber per change of tire vertical force (keyword = `CC_FZ_COEFFICIENT`). Sometimes called ride camber, this parameter is used to describe camber

resulting from structural deflection of the twist beam assembly under vertical load. This parameter is likely to have a small negative value.

Lateral Spacing

The construction of most twist beam suspensions resembles a pair of trailing links, connected laterally by a beam. The spring and damper components act on the trailing link portion of the structure. This arrangement makes the deflection of a spring as a function of vertical displacement of its associated wheel the same whether the vertical displacement results from two-wheel jounce or from roll, as with an independent suspension, so spacing information is not necessary. In some twist beam suspensions, the springs, dampers, and travel stops may act on the beam structure. With this arrangement the deflection of a spring for a given deflection of a wheel is less when the vertical deflection results from roll than when it results from two-wheel jounce.

Fields are provided to set parameters specifying the lateral spacing between springs, dampers, and jounce and rebound stops (25 - 28). When the spring and damper components act on the trailing link portion of the structure, these should be set to the track width. One way to do this is to use the symbolic name of the track width for the current axle: `L_TRACK`. Another is to put in the numerical value. Some advanced users can use these parameters to adjust for structural deformation in the beams that occur in K&C testing.

Libraries for Nonlinear Compliance, Springs, and Dampers

Table 3 lists the library screens provided to describe nonlinear properties of suspension systems and components, such as springs and shock absorbers. The root keywords are for the same Configurable Functions that are named at the start of the Spring and Damper section of the Echo file (Figure 22, page 40). Each screen has a Configurable Function with standard editing and viewing controls that are described in detail in the *VehicleSim Browser Reference Manual*. The Configurable Function datasets are identified for the VS Math Models by keywords based on the root names listed in the table.

A few of these functions underlie the spring and bump-stop screens that will be described in following subsections. The others have only standard editing and viewing controls, such that the information in Table 3 should be sufficient to document the screen.

Table 3. Summary of suspension table libraries.

Library Screen	Root Keywords	Description
Suspension: Spring	FS_COMP FS_EXT	Spring force vs. deflection
Suspension: Ride Rate (Spring + Tire)		Fz at ground vs. Z deflection at ground, with local tables RIDE_COMP and RIDE_EXT
Suspension: Shock Absorber	FD	Damper force vs. compression rate
Suspension: Auxiliary Roll Moment	MX_AUX	Auxiliary roll moment vs. suspension roll angle
Suspension: Measured Total Roll Stiffness		Roll moment at ground vs. roll angle at ground with local table MX_TOTAL
Suspension: Jounce and Rebound Stops	F_JNC_STOP F_REB_STOP	Jounce and rebound force vs. deflection

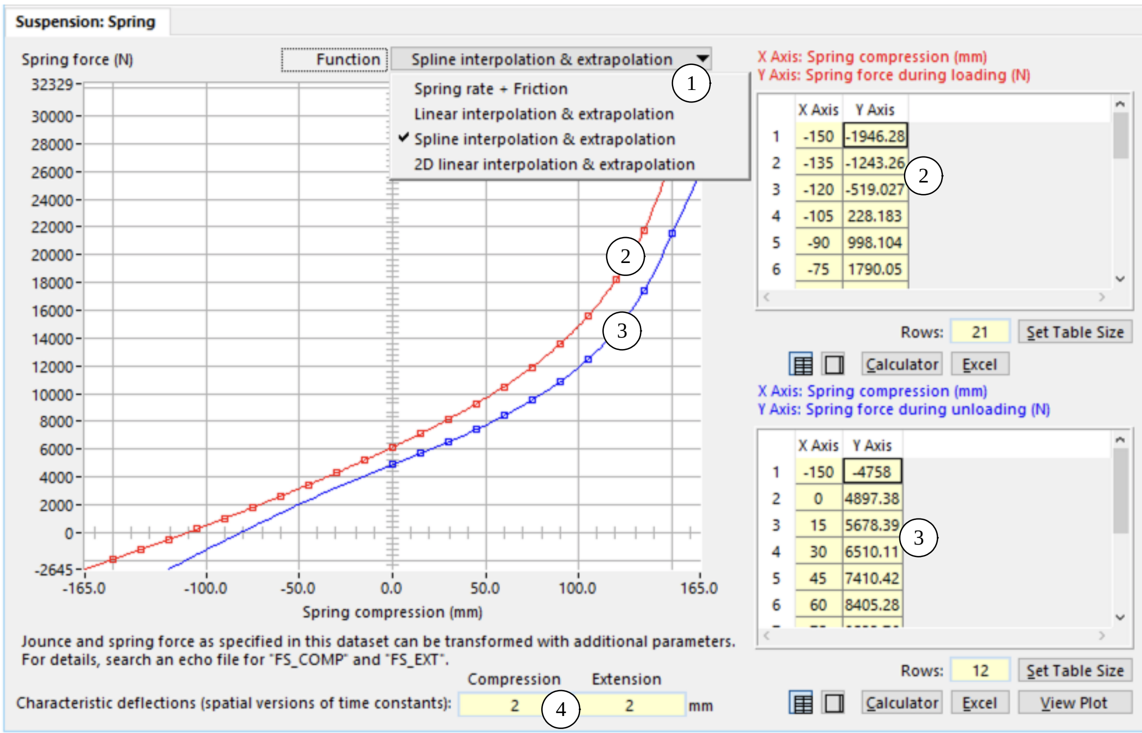
The relationships between suspension jounce and compression of a spring or damper are also represented internally with Configurable Functions. For each spring or damper compression function, there are two ways to configure the function. The first is to choose the linear version, in which case a yellow field appears on the screen and the keyword is the name of the function with the suffix _COEFFICIENT, e.g., CMP_SPR_SEAT_COEFFICIENT. The alternative is to define the compression as a function of the vertical movement at the wheel and link to a dataset from the **Generic Table** library. In this case, the function name must be specified to link the tabular data in the dataset to the function, e.g., CMP_SPR_SEAT. Table 4 lists the root keywords for the Configurable Functions used to calculate compression of the component from suspension compression at the wheel.

Table 4. Component compression functions.

Root Keyword	Compression
CMP_SPR_SEAT	Main suspension spring compression
CMP_DAMP	Shock absorber
CMP_JSTOP	Jounce bump stop
CMP_RSTOP	Rebound bump stop
CMP_SPR_SEAT_VIR	Main suspension spring compression, virtual steering axis suspension
CMP_DAMP_VIR	Shock absorber compression, virtual steering axis suspension
CMP_JSTOP_VIR	Jounce bump stop, virtual steering axis suspension.
CMP_RSTOP_VIR	Rebound bump stop, virtual steering axis suspension.

Suspension: Spring

The screen shown in Figure 32 is used to specify the force-displacement properties of the suspension springs, using the hysteretic VS model described earlier (page 4).



	X Axis	Y Axis
1	-150	-1946.28
2	-135	-1243.26
3	-120	-519.027
4	-105	228.183
5	-90	998.104
6	-75	1790.05

	X Axis	Y Axis
1	-150	-4758
2	0	4897.38
3	15	5678.39
4	30	6510.11
5	45	7410.42
6	60	8405.28

Figure 32. Suspension: Spring screen.

User Controls

You can choose between four methods for defining the basic force vs. deflection curves with a drop-down control (1). In the simplest case, the upper and lower curves are defined with a slope and a friction value defined as ½ the force separating the two curves, as was described earlier (Figure 24, page 42).

The spring screen shown in Figure 32 has controls similar to other screens in the VS Browser with tabular data. In this case, there are two tables for defining the upper and lower force/deflection curves. There are also two beta parameters.

(2) Table field for envelope of spring compression force as a function of spring compression during the loading process. The root keyword for the table is `FS_COMP`. Indices `IUNIT`, `IAXLE`, and `ISIDE` are used to specify the unit (1 = lead, 2 = 1st trailer), axle, and side.

The data from this table is plotted with the color red.

(3) Table field for envelope of spring force as a function of compression during the unloading process. The root keyword for the table is `FS_EXT`.

The data from this table is plotted with the color blue.

The spring model would be numerically unstable if the upper envelope (loading) were ever less than the lower envelope (unloading). This is also true when the tables are extrapolated. To guard against this potential source of error, the VS Math Models check the spring data during initialization, and will stop with an error message if there is a violation. There are four checks made of the data.

1. All values of compressive force in the table must be higher than the values of extension force that are calculated for the deflections used in the compression table.
2. All values of extension force in the table must be lower than the values of compressive force that are calculated for deflections used in the extension table.
3. The first gradient (spring rate) for the upper envelope must not be greater than the first gradient for the lower envelope, to ensure that extrapolated forces calculated for deflections less than the ranges covered in the tables are compatible.
4. The last gradient (spring rate) for the upper envelope must not be less than the last gradient for the lower envelope, to ensure that extrapolated forces calculated for deflections higher than the ranges covered in the tables are compatible.

4 Beta parameters. One of these parameters handles the transition up to the compression curve, and the other handling the transition down to the extension curve (shown between the vertical red lines in Figure 3). (Keywords = `SPRING_EXT_BETA` and `SPRING_COMP_BETA`).

As noted earlier, these are approximately 1/3 the distance needed to travel through a spring hysteresis (friction) loop.

Sensitivity to Static Load

Some springs (air springs in particular) have rate curves that change with the static load on the spring. These multiple spring curves can be specified using the 2D linear interpolation option with the drop-down control 1.

In the particular case of air springs, the pressure in the spring is altered to set the vehicle at a specific trim height regardless of load, preventing large variations in the attitude of the vehicle between empty and laden conditions. The force in the spring is proportional to pressure, and the change in pressure as the spring deflects is proportional to the change in volume due to deflection. The higher initial pressure associated with higher static loading thus produces a higher rate (larger changes in force per unit deflection) for the spring.

Air systems on trucks typically use a simple control valve to adjust air pressure as static load changes. TruckSim and CarSim use the load at initialization of the run, including payloads, to determine the spring characteristics to be used. Also, many air spring installations connect the left and right side springs so that they use a common air volume, hence the same pressure in both springs. When springs are connected like this, they don’t contribute to the roll stiffness. Springs in TruckSim and CarSim always contribute to roll stiffness, so you must define an auxiliary roll moment that removes the spring effects when simulating an interconnected air system.

Suspension: Ride Rate (Spring + Tire)

The **Suspension: Ride Rate (Spring + Tire)** screen (Figure 33, Figure 34) is an alternative to the **Suspension: Spring** screen described in the previous subsection, intended to be compatible with a popular measurement setup for vehicle ride rate.

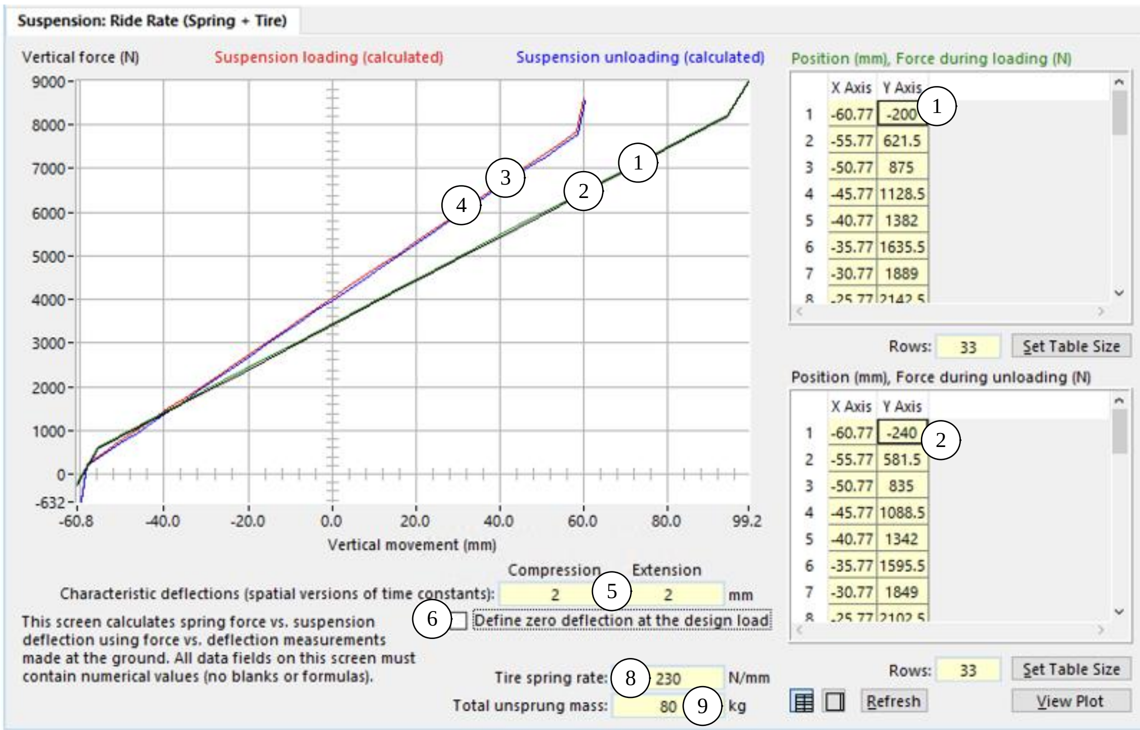


Figure 33. The Suspension: Ride Rate (Spring + Tires) screen.

Discussion

Spring properties are sometimes measured as installed in the vehicle by fixing the vehicle body (sprung mass) in a laboratory and moving the wheels up and down with vertical actuators under the tires. In this test condition, changes in vertical tire force are due to compression of both the tire and the suspension spring. This screen performs calculations automatically to remove the tire effects, leaving the nonlinear spring behavior.

Note	When using this screen, you are actually specifying the *wheel rate* (change in force at the wheel center for a change in suspension compression), not the *spring rate*. This means the mechanical ratio for the springs should be set to one for any suspension dataset that links to a dataset from this library.

This screen has similar controls to the basic spring screen, but with the following differences:

1. The tables of force vs. position values are measured at the ground (under the tires), rather than for the spring in isolation.

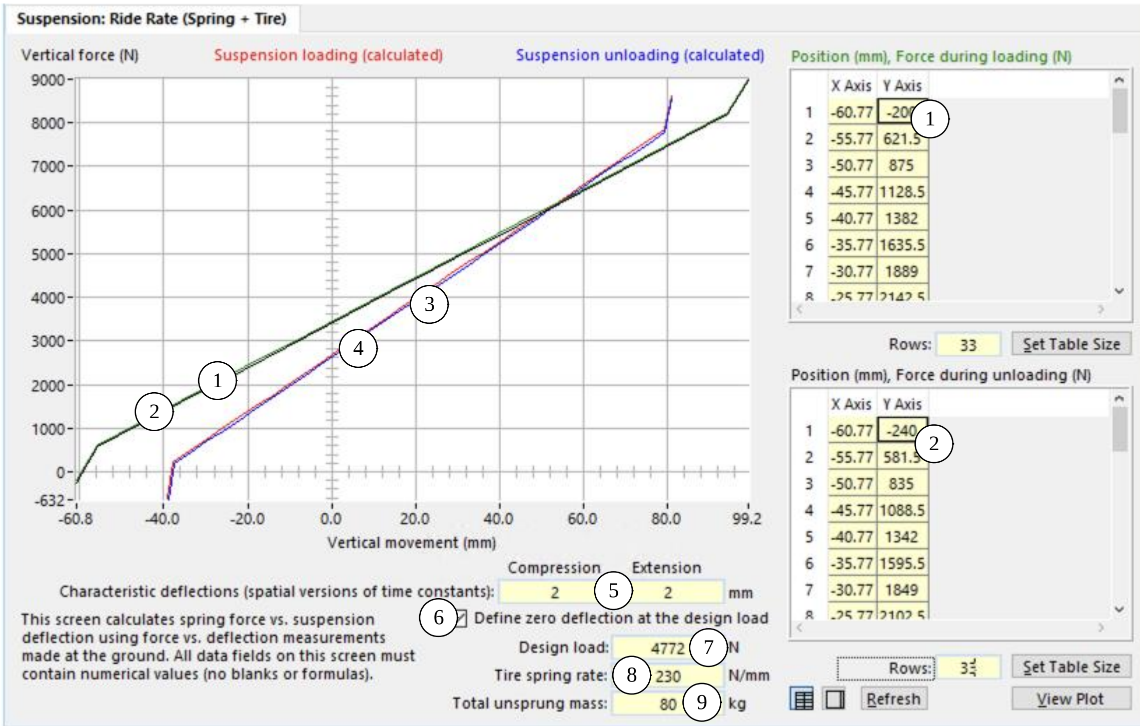


Figure 34. Ride Rate screen with zero deflection specified.

2. The screen does not have a control to specify a form of interpolation: it will create a table with linear interpolation to send to the vehicle VS Math Model.
3. It includes three extra parameters (tire spring rate 8, unsprung mass 9, and optionally design load 8, Figure 34), used to calculate the spring properties from the measured movement and force at the tire/actuator contact.

This screen has a graph showing four plots (Figure 33, Figure 34). The two plots shown in the colors green and black (1 and 2) correspond to measurements of tire force (F_{ride}) in response to the vertical movement of the actuators under the tires (X_{ride}). The two shown in the colors red and blue (3 and 4) are calculated by removing the tire effects, and correspond to the behavior of the suspension springs alone (X_{spring} and F_{spring}), such as:

$$X_{spring} = X_{ride} - (F_{ride} / k_t) \tag{1}$$

$$F_{spring} = F_{ride} - (M_{us} / 2) * 9.80665 \tag{2}$$

where k_t is the tire spring stiffness and M_{us} is the total unsprung mass (both wheels).

By using the above equation (1), the suspension spring deflection (X_{spring}) won't be zero at the *design load*. There is an alternative option to define the zero spring deflection at the design load (checking the checkbox 6), by using the following equation instead of equation (1):

$$X_{spring} = X_{ride} - (F_{ride} - F_{design}) / k_t \tag{3}$$

where F_{design} is the design load for each wheel side.

The numbers for the springs alone are stored in the associated Parsfile and can be viewed with the text editor. However, they are not shown on this screen because they are re-calculated whenever a change is made to any of the information on the screen.

You should view the red and blue plots to confirm their validity. If tire properties that are specified are not representative of the properties as they existed when the measurements were made, the calculated points in the red and blue plots can show unusual results, such as a negative spring rate.

User Settings

1. Table field for envelope of measured ground force as a function of compression (keyword = RIDE_COMP_ENVELOPE). This represents the force vs. deflection that would be measured while the spring is moving in compression (jounce).

   The values in this table are used, along with other properties specified on the screen, to define spring compression force as a function of spring compression. The calculated numbers are plotted with red line 3 and stored in the same Parsfile (keyword = FS_COMP).

2. Table field for envelope of measured ground force as a function of extension (keyword = RIDE_EXT_ENVELOPE). This represents the force vs. deflection that would be measured while the spring is moving in the extension (rebound) direction.

   The values in this table are used, along with other properties specified on the screen, to define spring force as a function of spring extension. The calculated numbers are plotted with a blue line 4 and stored in the same parsfile (keyword = FS_EXT).

   The spring model would be numerically unstable if the upper envelope (compression) were ever less than the lower envelope (extension). This is also true when the tables are extrapolated. To guard against this potential source of error, the VS Math Models check the spring data during initialization, and will stop with an error message if there is a violation. The checks made by the VS Math Models are described in the previous subsection (page 59).

3. Plot of spring force as a function of compression during loading. The values used to make this plots are updated whenever a change is made on the screen. They will be used in the VS Math Model and will appear in the Echo file. They are also in the Parsfile for the screen (keyword = FS_COMP).

4. Plot of spring force as a function of compression when uloading. The values used to make this plots are updated whenever a change is made on the screen. They will be used in the VS Math Model and will appear in the Echo file. They are also in the Parsfile for the screen (keyword = FS_EXT).

5. Beta parameters for compression and extension (keywords = SPRING_EXT_BETA and SPRING_COMP_BETA). These are approximately 1/3 the distance needed to travel through a spring hysteresis (friction) loop.

6. Checkbox to define the zero suspension deflection at the design load. When checked, a data field is shown that specifies the design load for each wheel side 7 (Figure 34) and equation (3) is used to calculate the spring deflection. If not checked (Figure 33), the yellow field for design load is hidden and equation (1) is used instead of equation (3). The force information for the spring is not affected by this option, but the compression data for the spring is affected.

The option exists for cases where there is a need to compared spring compression in the math model to spring compression data from the ride test.

7 Design load at the ground for the current side of the vehicle. This design load is used to convert the measured ground force to a spring force only when the checkbox 6 is checked.

8 Tire spring rate for the tire that was on the vehicle when the ground force was measured. You can use a different tire spring rate on the vehicle for your simulation runs—this coefficient is used only to convert the measured ground force to a spring force.

If the suspension had dual tires, the spring rate should be combined for both tires on the wheel.

9 Unsprung mass (both wheels) of the suspension as it was on the vehicle when the ground force was measured. You can use a different unsprung mass on the vehicle for your runs — this mass is used only to convert the measured ground force to a spring force.

Bump Stops

Bump stops can be incorporated into the spring table or defined using the separate **Suspension: Jounce and Rebound Stops** tables. If you include the force due to suspension stops in the spring table, any tandem axle load sharing effects (TruckSim or CarSim Trailer) also distribute the suspension stop forces. In most tandem suspensions, the load sharing mechanism does not distribute the bump stop forces, so using the **Suspension: Jounce and Rebound Stops** tables is preferred.

When using the **Suspension: Jounce and Rebound Stops** tables, imagine a spring at the location of the stop. The spring produces zero force for some deflection, until the stop is contacted. Then the spring produces force according to the stiffness of the stop. The deflection over which the force is zero is sometimes called *engagement travel* or *engagement length.*

Auxiliary Roll Stiffness

The VS Math Models include a Configurable Function `MX_AUX` to provide an auxiliary roll moment to account for differences between the roll stiffness due only to springs and their spacing, and the roll stiffness that might be measured for the suspension. This function provides a moment from a torsional spring connecting a solid axle to the sprung mass, or a set of matched suspension forces for an independent suspension to provide a torsional spring connecting a virtual axle defined by connecting the wheel centers on both sides of the suspension.

Note Although it is called a torsional spring for convenience, the auxiliary roll moment in the math model can be due to other factors such as twisting of linkages, bending of sheet metal, binding of linkages, etc. The auxiliary roll moment accounts for all roll moment effects other than the main suspension springs.

In some cases, the measured roll moment for a suspension is less than the moment applied by the springs at the specified lateral spacing. In cases where the auxiliary moment is accounting for additional compliances, it will have a negative slope.

The library **Suspension: Auxiliary Roll Moment** is available to provide datasets for the function using the same user interface provided for other Configurable Functions.

Suspension: Measured Total Roll Stiffness screen

Another library, **Suspension: Measured Total Roll Stiffness**, is available as an alternative for users with laboratory measurements of the overall suspension roll stiffness (Figure 35).

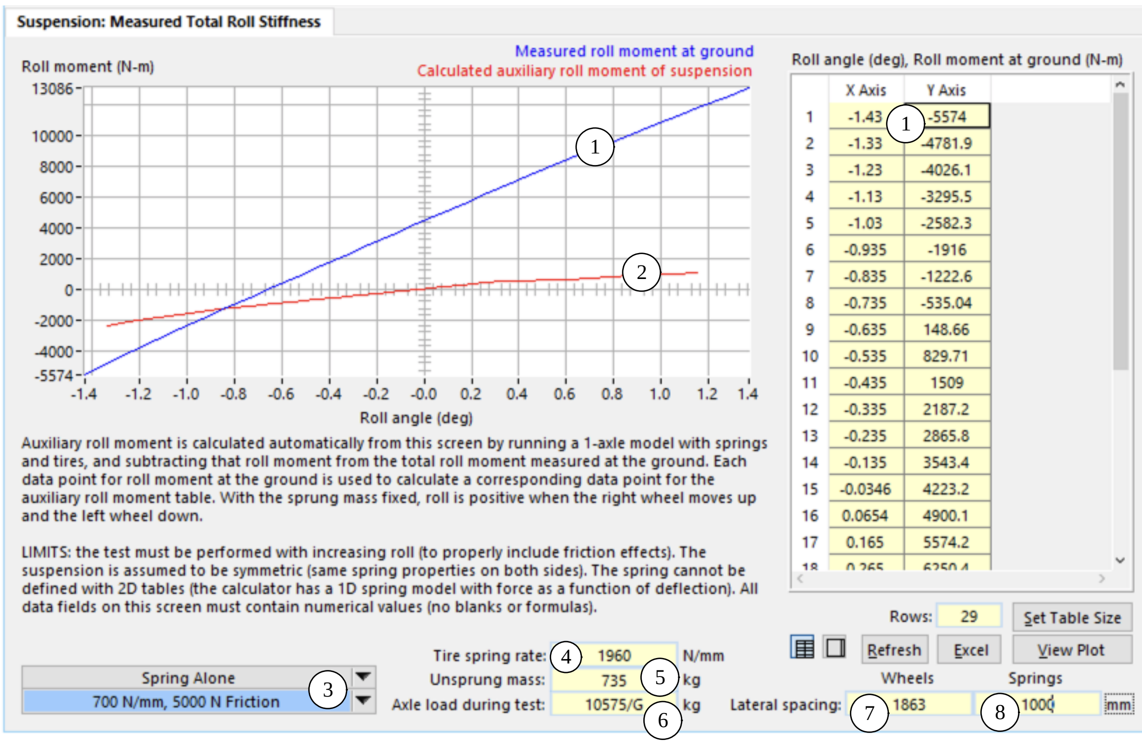


Figure 35. The Suspension: Measured Total Roll Stiffness screen.

A physical measurement of this type is typically made by fixing the vehicle body and rolling the "ground" under the tires, where the "ground" is defined by vertical hydraulic actuators. To provide roll, the actuator on one side is raised while the actuator on the other side is lowered. The actuators are typically controlled to maintain a constant total vertical force and zero lateral force during the test. The vertical movements of the actuators are combined with their lateral spacing to define a "ground angle." The vertical force measured at each actuator is combined with the lateral spacing to define a "ground roll moment."

This screen shows two plots. The first, in the color blue 1, shows the measured data listed in the table 1. The second, in the color red 2, shows the auxiliary roll moment calculated automatically from the measured data and the other information on the screen. The plot shown in red illustrates the only data values from this screen that are actually transferred to the VS Math Model. The numbers are stored in the associated Parsfile and can be viewed with the text editor. (They are not shown on this screen because they are re-calculated whenever a change is made to any of the information on the screen.)

View the red plot to determine the validity of the measured data and associated suspensions and tire properties. If the spring or tire properties that are specified are not representative of the properties as they existed when the measurements were made, the calculated points in the red plot can show unusual results, such as a negative spring rate.

The table of auxiliary moment vs. roll of the sprung mass relative to the axle will have data points calculated for the same conditions represented in the table that relates moment and angle at the ground (1). The calculations for the auxiliary roll moment dataset are made with a VS Math Model that is described in a following subsection for the benefit of advanced users (page 67).

> **Alert** Because the table of points for the `MX_AUX` function is calculated with a separate VS Math Model, it is essential that all yellow fields be given values ((4) - (8)), and that the blue link (3) be connected to a spring dataset that does not use the 2D Carpet Plot option.

User Settings

(1) Two-column table of values of measured roll moment applied from the vehicle to the ground as a function of vehicle roll relative to the ground. A positive ground angle (where the label "roll angle" is short for "vehicle roll relative to the ground) occurs when the right wheel is higher than the left wheel. This table is identified in the Parsfile for the dataset with the keyword `MX_TOTAL_TABLE`.

The values in this table are used, along with other properties specified on the screen, to define auxiliary roll moment as a function of suspension roll (2).

(2) For each row in the table of total ground roll and total roll moment, a value is calculated for suspension roll (relative) and auxiliary roll moment. These values are plotted in the color red, and are written in the Parsfile for the dataset with the keyword `MX_AUX_TABLE`. This keyword is processed by the VS Math Model for the vehicle to obtain a dataset for the `MX_AUX` Configurable Function for the current suspension.

(3) Link to a **Suspension: Spring** dataset or a **Suspension: Ride Rate (Spring + Tire)** dataset for the springs that were on the vehicle when the roll moment was measured. You can use a different spring for your runs—the spring data from this link is used only to convert the measured ground roll moment to an auxiliary roll moment.

The calculated moments are based on the assumption that the spring has a mechanical advantage of 1, such that 1 mm of suspension travel corresponds to 1 mm of spring deflection. If the spring data has significant nonlinearities, then be sure the table of total moment vs. ground angle has enough data points to generate the proper nonlinear relationship.

> **Alert** As noted above, the calculator tool uses 1D Configurable Functions for the force vs. deflection behavior of the spring for loading and unloading.
>
> If the spring dataset for the suspension has 2D data, you can make a copy with 1D data that applies for the nominal spring load used during the roll moment test, and then link to that copy from this screen.

4 Tire spring rate for the tires that were on the vehicle when the roll moment was measured. You can use a different tire spring rate for your runs —this coefficient is used only to convert the measured ground roll moment to an auxiliary roll moment.

The spring rate applies for each wheel. If the suspension had dual tires, the rate that is provided for the wheel should be for the combined stiffness of the two tires.

5 Unsprung mass (both wheels) of the suspension as it was on the vehicle when the ground force was measured. You can use a different unsprung mass for your runs—this mass is used only to determine the spring loads at zero roll during the measurement.

6 Axle load during test. Specify the mass supported by the tires during the test. This is used to determine the spring loads during the test and therefore use the proper nonlinear range of the spring tables. Note that the load is specified in kg. If you have the load in Newtons, you can divide by G to provide the kg value, as shown in Figure 35.

7 Lateral spacing of wheel centers (track width) when the roll moment was measured. You can use a different track width for your runs—this track width is used only to determine the moment arm associated with the tire forces.

8 Lateral spacing of springs when the roll moment was measured. For a solid-axle suspension, specify the physical distance between the springs where they attach to the axle. For an independent suspension, use the wheel track width (same as 7).

The Maux VS Math Model

The calculations made to obtain a `MX_AUX` dataset are handled by a VS Solver library `maux.dll` that contains a model with a fixed sprung mass, and single moving body (a solid axle), tire springs, and nonlinear suspension springs with hysteresis. This subsection is intended for advanced users interested in learning more about the model.

Input and output files for the Maux model are written in the library folder for this screen. As with any library, view the content with the **View** menu item **Open this Library Folder in Windows**. For example, Figure 36 shows a folder containing the example dataset from Figure 35. The Parsfile for the datasets has the UUID file name with the extension `.par` 3. There is an echo file 4, a log file 5, and two output files with data for plotting (with extensions `.erd` and `.bin`) 2. The files are specified in a simfile that is updated whenever a change is made to a dataset 1.

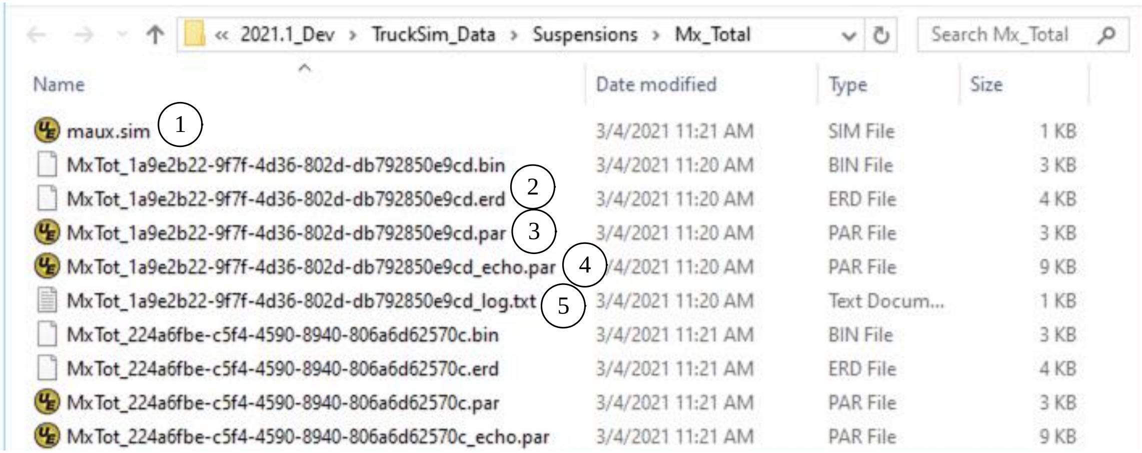


Figure 36. Library folder for a TruckSim example database.

The text files (with extensions `.par` and `.txt`) may be viewed with any text editor. For example, Figure 37 shows part of the Echo file generated by the Maux math model for the dataset shown earlier (Figure 35).

> **Note** The keywords used in the Maux math model are distinct from those used in CarSim and TruckSim, to avoid interfering with settings for the full VS Math Model for the vehicle.

The table for the calculated values in the `MX_AUX` table are not listed in the Echo file. The VS Browser communicates with the Maux model to obtain the values for each point, and writes them into the dataset Parsfile that can be viewed using the Parsfile button on the library screen.

The Maux math model calculates many other variables for each combination of angle and roll moment. These may be viewed with VS Visualizer as follows:

1. On the screen for this library, click the button **View Plot** button 4 (Figure 38). VS Visualizer will launch, and show the same two plots as seen on the library screen (1 and 2).
2. Use the **File** menu item **Open** 3, and select the ERD file for the run (e.g., 2 in Figure 36). This will cause VS Visualizer to show a window pane listing all of the variables in the newly loaded files 1 (Figure 39).
3. Use the **Plot** menu 2 item **Create New Plot** to create a blank window pane for making a new plot 3. There are two ways to select data to plot:
    a. Drag names from the **Data Manager** window into the plot area. In the figure, this was done to plot the four variables for spring 4 and tire 5 forces.
    b. Right-click in the plot area to obtain a pop-up menu. Choose the item **Add data to plot**. This option allows data channels to be specified for both the X and Y axes.

Note Please see the *VS Visualizer Reference Manual* for more information about making plots interactively.

MxTot_1a9e2b22-9f7f-4d36-802d-db792850e9cd_end.par

```
TSTEP          0.0005 ; s ! [D] Time step for numerical integration [L]
! TSTEP_WRITE  0.0005 ; s ! CALC -- Time interval in output time-series file
! T_DT         0.0005 ; s ! CALC -- Time increment between calculations

!------------------------------------------------------------------------------
! MODEL PARAMETERS
!------------------------------------------------------------------------------
F_STATIC  1078.349895 ; kg ! Total (left + right) tire load [I]
KTIRE            1960 ; N/mm ! Vertical spring rate for one tire [I]
LSPRING          1000 ; mm ! Distance between suspension springs [I]
LTRACK           1863 ; mm ! Track width [I]
MUS               735 ; kg ! Unsprung mass of axle/suspension (both sides) [I]

!------------------------------------------------------------------------------
! SYSTEM CONSTANTS
!------------------------------------------------------------------------------
! DR      57.29577951 ; - ! Deg/rad symbol to use in formulas (read-only)
! G           9.80665 ; - ! Symbol for gravity constant m/s/s (read-only)
! PI      3.141592654 ; - ! Symbol for PI to use in formulas (read-only)
! ZERO              0 ; - ! Symbol for zero to use in VS Events (read-only)

!------------------------------------------------------------------------------
! CONFIGURABLE FUNCTIONS
!------------------------------------------------------------------------------

! FS_COMP: Compression (upper) envelope of suspension spring force (1 side) as a
! function of deflection. Spring force is a function of deflection (COEFFICIENT or
! TABLE). Spring force from the calculation can be adjusted with FS_COMP_GAIN and
! FS_COMP_OFFSET. An inverse version of this function is used internally. The
! derivative of this function is used internally.

FS_COMP_COEFFICIENT 700 ! Coefficient: spring force per unit deflection (N/mm)
FS_COMP_GAIN          1 ! Gain multiplied with calculated value to get spring force
FS_COMP_OFFSET     5000 ; N ! Offset added (after gain) to get spring force

! FS_EXT: Extension (lower) envelope of suspension spring force (1 side) as a
! function of deflection. Spring force is a function of deflection (COEFFICIENT or
! TABLE). Spring force from the calculation can be adjusted with FS_EXT_GAIN and
! FS_EXT_OFFSET. An inverse version of this function is used internally. The
! derivative of this function is used internally.

FS_EXT_COEFFICIENT 700 ! Coefficient: spring force per unit deflection (N/mm)
FS_EXT_GAIN          1 ! Gain multiplied with calculated value to get spring force
FS_EXT_OFFSET    -5000 ; N ! Offset added (after gain) to get spring force

! MX_TOTAL: Total roll moment applied to axle via tires as a function of ground
! roll. Column 1 = roll (deg). Column 2 = moment (N-m).. Moment is a function of roll
! angle (CONSTANT, COEFFICIENT, or TABLE). Moment from the calculation can be
! adjusted with MX_TOTAL_GAIN and MX_TOTAL_OFFSET. The derivative of this function is
! used internally.

! 1D table: col 1 = roll angle (deg), col 2 = moment (N-m)
MX_TOTAL_TABLE LINEAR ! linear interpolation and extrapolation
 -1.43, -5574
 -1.33, -4781.9
 -1.23, -4026.1
```

Figure 37. Part of Echo file written by the Maux math model.

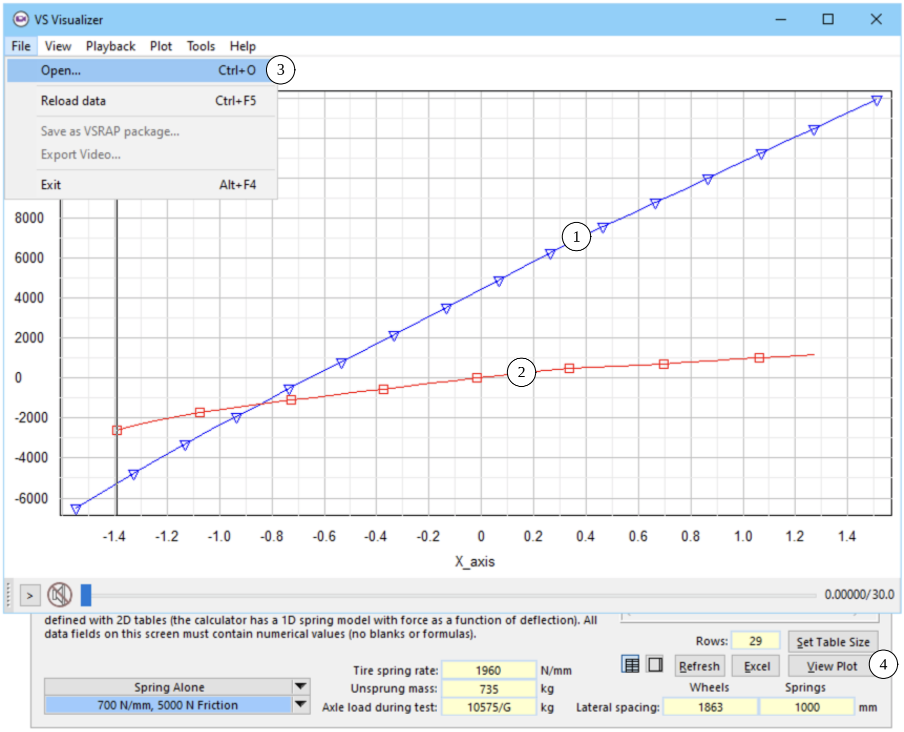


Figure 38. View plots with VS Visualizer.

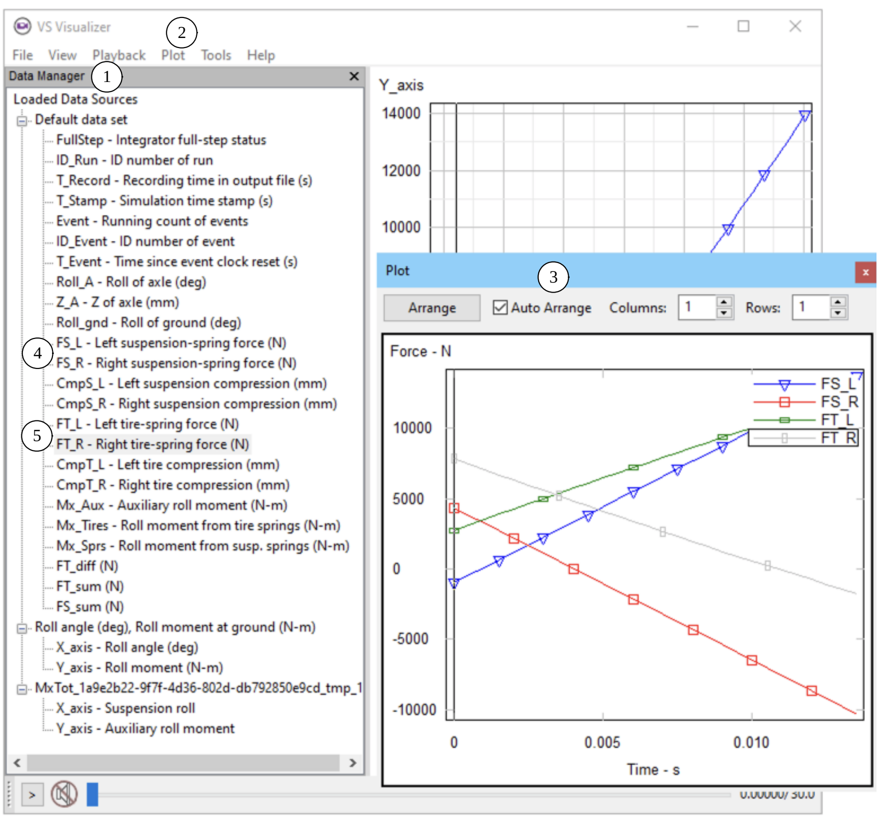


Figure 39. New panes for loaded data channels and a new plot.

Suspension: Compliance (Nonlinear)

As mentioned earlier, the top-level screens for assembling data for suspension compliance, springs, and dampers include a drop-down control to choose the source of data for the compliances. If nonlinear compliance data are available, a link can be made to the screen shown in Figure 40.

Each compliance effect is represented on the screen with a drop-down control 1 used to specify whether the compliance function should be configured to use a linear coefficient (as done on the three compliance assembly screens described in previous subsections) or a blue link to a dataset, from the **Generic Table** library.

As with other suspension screens, it has a checkbox 2 to specify whether separate properties are used for the left and right sides. If not checked, each compliance setting is copied for both sides. This screen can be used with any type of suspension. A drop-down control 3 is used to specify the type of suspension. Based on the selection, it shows only the applicable compliance effects.

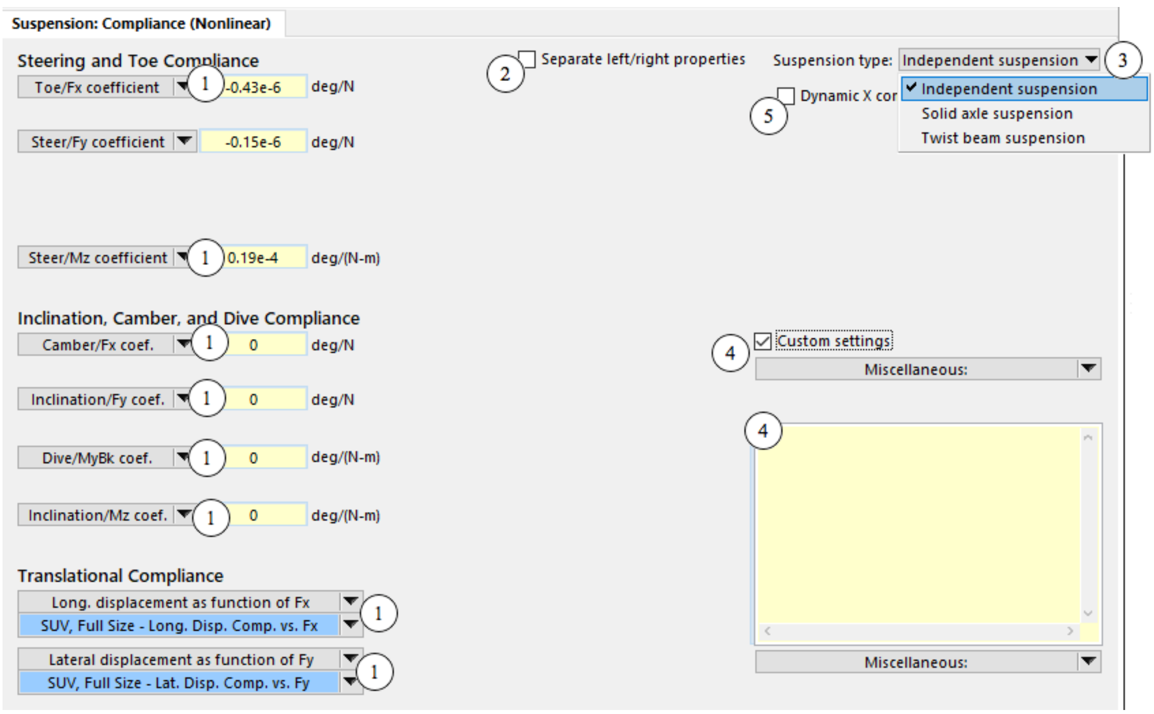


Figure 40. The Suspension: Compliance (Nonlinear) screen.

Note For the CarSim virtual steering axis suspension, select the "Independent suspension" option in control 3, and do not use the dynamic X compliance option 5, as that is only for independent suspensions.

Table 5 lists the Configurable Functions available to describe suspension compliance effects. For each effect in the table, there are two ways to specify the effect. The first is to choose the linear version, in which case a yellow field appears on the screen (e.g., 1) and the keyword is the name of the function with the suffix `_COEFFICIENT`, e.g., `CT_FX_COEFFICIENT`. The alternative is to define the compliance deflection as a function of the force or moment and link to a dataset from the **Generic Table** library. In this case, the root name of the function must be specified to link the tabular data in the dataset to the compliance function, e.g., `CT_FX`.

Nine compliances can be extended to 2D tables with an additional independent variable of the tire force on the opposite side of the axle. This is an alternative way to express the twist beam suspension; details are described in a separate technical memo: *Twist Beam Model with 2D Tables*. `C_LONG_AXLE` and `C_LAT_AXLE` are the two translational compliances for solid axle suspensions.

When the checkbox **Custom settings** 4 is checked, a link and a miscellaneous field are shown to support additional settings for advanced users (VS Commands, etc.)

Table 5. Summary of suspension compliance Configurable Functions.

Type	Root Name	2D?	Description
All	`CT_FX`	Yes	Toe angle deflection due to Fx
All	`CS_FY`	Yes	Steer angle deflection due to Fy
Twist beam	`CT_FZ`	No	Toe angle deflection due to Fz
All	`CS_MZ`	Yes	Steer angle deflection due to Mz
All	`CC_FX`	Yes	Camber angle deflection due to Fx
All	`CI_FY`	Yes	Inclination angle deflection due to Fy
Twist beam	`CC_FZ`	No	Camber angle deflection due to Fz
Independent, virtual steering axis	`CD_MY`	Yes	Dive angle deflection due to braking torque
All	`CI_MZ`	Yes	Inclination angle deflection due to Mz
Independent, twist, virtual steering axis	`C_LONG`	Yes	Longitudinal deflection due to Fx
Solid axle	`C_LONG_AXLE`	No	Longitudinal deflection due to Fx (axle)
Independent, twist, virtual steering axis	`C_LAT`	Yes	Lateral deflection due to Fy
Solid axle	`C_LAT_AXLE`	No	Lateral deflection due to Fy (axle)

For independent suspensions in CarSim only, the **Dynamic X compliance** checkbox 5 can be used to activate the dynamic longitudinal compliance described earlier (Dynamic X compliance for independent suspensions, p. 37). When the box is checked, the longitudinal compliance yellow field switches to math model keyword `FX_SUSP_CMP_COEFFICIENT`, as the quasi-static compliance setting `C_LONG_COEFFICIENT` is no longer needed. You can also link to a dataset from the **Generic Table** library. In this case, the root name of the function, `FX_SUSP_CMP`, must be specified in the generic table dataset. If non-default values are needed, the custom settings area 4 can be used to set the damping associated with the dynamic X compliance, math model keyword `FX_SUSP_DMP`.