Mechanical Simulation

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

Example: Self-Steer Axle

David E. Kline, Ph.D.

Senior Modeling & Simulation Engineer

Introduction

Self-steer axles are used to reduce tire scrub, often as auxiliary axles that can be raised or lowered depending on the payload being carried or bridge weight limits on the route. A typical application is a cement or dump truck, with one or more self-steer axles placed ahead of the tandem drive axles (Figure 1). The TruckSim steering system readily supports *force*-steer axles, where the steer of the axle is actively controlled. Self-steer axles are possible to include in TruckSim using the steering system import variables. This approach requires a separate steering system model that can be used to calculate the values imported for use with TruckSim.

Figure 1. Dump truck with self-steer axle positioned ahead of tandem drive axles.

External Self-Steer Model

TruckSim does not have a native self-steer model but supports options for user-defined model extensions through external software or the built-in VS Command scripting language. This document describes a self-steer model that has been added using VS Commands.

Figure 2 shows a symmetric self-steer mechanism that interconnects the left and right wheels with a rigid tie rod. The tie rod is assumed to be connected at a distance h behind the kingpins. Each side has a steering stabilizer mounted at an angle α with respect to the axle's long axis. The stabilizers are modeled as linear dampers with rate c.

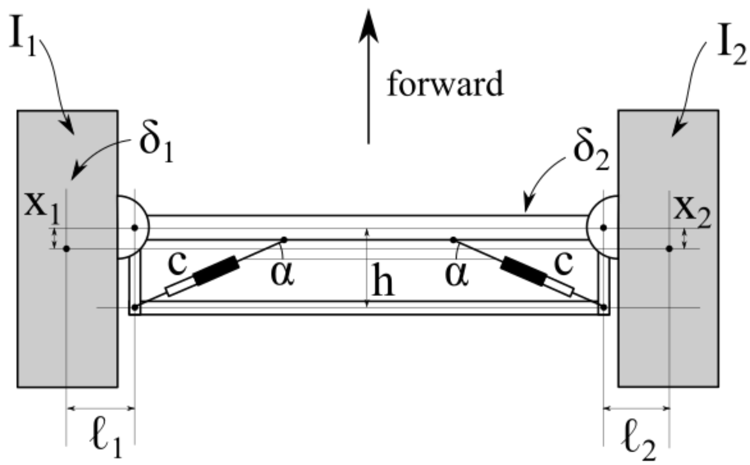

Figure 2. Self-steer axle model used in this example.

Each wheel is assumed to have an inertia about its kingpin (I_1 left, I_2 right), while the kingpin itself is modeled as vertical but offset by a longitudinal value (x_1 left, x_2 right) and a lateral value (ℓ_1 left, ℓ_2 right). The steer of the left wheel about its kingpin is denoted by δ_1, while that of the right wheel is δ_2. Because the kingpins are vertical, these angles are equivalent to the steer angles of the wheels.

The rigid tie rod establishes a kinematic constraint between the left and right wheels of the axle. For the purposes of this example, this kinematic constraint is idealized as making the steer of the left wheel equal to the steer of the right wheel:

$$\delta_1 = \delta_2.$$

Assume that the forces and moments on each tire can be resolved into net kingpin moments τ_1 and τ_2, for the left and right wheels, respectively. The equation of motion for the left wheel is then

$$I_1\ddot{\delta}_1 = \tau_1 - c\dot{\delta}_1 h \cos \alpha.$$

Similarly, the equation of motion for the right wheel is

$$I_2\ddot{\delta}_2 = \tau_2 - c\dot{\delta}_2 h \cos \alpha.$$

Adding the left and right equations of motion together and applying the kinematic constraint gives the equation of motion for the external steering system:

$$(I_1 + I_2)\ddot{\delta}_1 = \tau_1 + \tau_2 - 2ch\dot{\delta}_1 \cos \alpha.$$

Using the External Model with TruckSim

The steering model option for each suspension is set using the indexed parameter `OPT_STEER_EXT`. For axle 2, the option is selected to use a full external model with the setting:

```
OPT_STEER_EXT(2) = 4
```

This tells the VS Math Model that while the second axle is steered, do not use any of the internal calculations to calculate how much the wheels are steered. Rather, the steer angles and the steer rates of the wheels are assumed to be calculated externally and supplied to the math model with the following import variables:

- `IMP_STEER_L2`. The steer angle of the left wheel of the second axle.
- `IMP_STEER_R2`. The steer angle of the right wheel of the second axle.
- `IMP_DSTEER_L2`. The steer rate of the left wheel of the second axle.
- `IMP_DSTEER_R2`. The steer rate of the right wheel of the second axle.

Importantly, the kingpins still exist within the VS Math Model. The import variables supply the rotation and speed of the wheels, which are equivalent to the rotation and speed about the kingpins considering the assumption of a vertical kingpin axis.

The calculations are accomplished using the external model derived in the previous section. First consider the parameters used by the model.

- Steer arm length h. Set with user-defined parameter `SSA_LEN` [mm].
- Stabilizer mounting angle α. Set with user-defined parameter `SSA_ANG` [deg].
- Stabilizer damping c. Set with user-defined parameter `SSA_DMP` [N-s/mm].
- Left kingpin longitudinal offset x_1. Set with TruckSim parameter `X_KPO(2,1)` [mm].
- Right kingpin longitudinal offset x_2. Set with TruckSim parameter `X_KPO(2,2)` [mm].
- Left kingpin lateral offset ℓ_1. Set with TruckSim parameter `L_KPO(2,1)` [mm].
- Right kingpin lateral offset ℓ_2. Set with TruckSim parameter `L_KPO(2,2)` [mm].
- Inertia of left side about kingpin I_1. Set with user-defined parameter `SSA_ILS` [kg-m2], but this parameter is calculated from TruckSim parameters per below discussion.
- Inertia of right side about kingpin I_2. Set with user-defined parameter `SSA_IRS` [kg-m2], calculated according to below discussion.

For the two inertias, the inertia about the kingpin is assumed to be coming from the mass and yaw inertia of the steered portion of the unsprung mass. Considering the left wheel of the second axle, the total steered unsprung mass is given by

```
M_US_STR(2,1) + M_TIRE(2,1,1).
```

The total yaw inertia of the same wheel is

```
IW_XXZZ(2,1) + IT_XXZZ(2,1,1).
```

The total inertia about the kingpin for the left is then

```
(M_US_STR(2,1) + M_TIRE(2,1,1))*(L_KPO(2,1)^2 + X_KPO(2,1)^2) +
                    IW_XXZZ(2,1) + IT_XXZZ(2,1,1)
```

when considering the distance from the kingpin to the wheel center. The value for the right side can be calculated similarly.

Aside from the parameter values, the net kingpin moments are also needed. These can be accessed using the output variables `Mz_KP_L2` and `Mz_KP_R2`.

Recall the equation of motion for the external steering system. This is rewritten as

$$\ddot{\delta}_1 = \frac{\tau_1 + \tau_2 - 2ch\,\dot{\delta}_1 cos\,\alpha}{I_1 + I_2}.$$

This equation is integrated twice, at the end of the current timestep, using VS Command `EQ_DIFFERENTIAL`, to get the values needed for the import variables. The steer rate and moments on the right-hand side are thus from the current timestep. The integrated values are then applied at the beginning of the next timestep using the VS Command `EQ_IN`.

VS Commands are listed at the end of Echo files. Figure 3 shows the relevant part of the Echo file for a low-speed figure-eight example of the self-steering model for axle 2. (Only the first two import statements are shown, to save space.)

The dump truck with self-steer axle (Figure 1) is implemented by adding the appropriate self-steer axle datasets to the three-axle dump truck. This is done by creating a new four-axle lead unit vehicle assembly with the appropriate dataset links (Figure 4). The self-steer axle's kinematics, compliance, and brakes are assumed representable by some of the example datasets. On the other hand, the external steering model is a unique dataset, **Self-Steer Axle VS Commands**, implementing the VS Commands covered in this section.

The dump truck with self-steer axle is exercised using several test procedures:

- Low speed figure-eight
- Three-point turn
- 100-0 km/h braking test
- Double lane change at 40 km/h

Another example shows how the self-steer axle can be raised and lowered using VS Commands.

Results and Discussion

The figure-eight example readily shows the improvement in turning radius and reduction in tire slip (Figure 5). The red truck, without the self-steer axle, has a maximum lateral path deviation greatly exceeding that of the blue truck, which has the self-steer axle (-2 meters instead of 9 millimeters). Moreover, axle 2 of the blue truck has slip angle magnitudes closer to 0.5 degrees rather than the 6 to 7 degrees seen with the non-steered axle.

```
!-------------------------------------------------------------------------------
! NEW VARIABLES DEFINED AT RUN TIME
!-------------------------------------------------------------------------------
DEFINE_PARAMETER SSA_OFF = 0; - ;
DEFINE_PARAMETER SSA_LEN = 300; mm ;
DEFINE_PARAMETER SSA_ANG = 10; deg ;
DEFINE_PARAMETER SSA_DMP = 10; N-s/mm ;
DEFINE_PARAMETER SSA_ILS = 11.43; kg-m2 ;
DEFINE_PARAMETER SSA_IRS = 11.43; kg-m2 ;

DEFINE_VARIABLE SSA_STR_RATE = 0; - ;
DEFINE_VARIABLE SSA_STR = 0; - ;

!-------------------------------------------------------------------------------
! EQUATIONS IN (AT THE START OF EVERY TIME STEP)
!-------------------------------------------------------------------------------
EQ_IN IMP_STEER_L2 = IF(SSA_OFF, 0, SSA_STR);
EQ_IN IMP_STEER_R2 = IF(SSA_OFF, 0, SSA_STR);
EQ_IN IMP_DSTEER_L2 = IF(SSA_OFF, 0, SSA_STR_RATE);
EQ_IN IMP_DSTEER_R2 = IF(SSA_OFF, 0, SSA_STR_RATE);

!-------------------------------------------------------------------------------
! DIFFERENTIAL EQUATIONS FOR NEW STATE VARIABLES (AT THE END OF EVERY TIME STEP)
!-------------------------------------------------------------------------------
EQ_DIFFERENTIAL SSA_STR_RATE = (MZ_KP_L2 + MZ_KP_R2 -(2*SSA_DMP*SSA_LEN*COS(SSA_ANG)*
AVZKP_L2))/(SSA_ILS + SSA_IRS);
EQ_DIFFERENTIAL SSA_STR = SSA_STR_RATE;

!-------------------------------------------------------------------------------
! IMPORTED VARIABLES, RELATIONS TO NATIVE VARIABLES, INITIAL VALUES, and UNITS
!-------------------------------------------------------------------------------
IMPORT IMP_STEER_L2 VS_REPLACE 0 ; deg ! #0. Road wheel L2 steer angle due to the
                                 ! steering system, (NOT ride/roll steer) from
                                 ! external model
IMPORT IMP_DSTEER_L2 VS_REPLACE 0 ; deg/s ! #0. Road wheel L2 steer angular rate due
```

Figure 3. Echo file showing VS Commands that define self-steering for axle 2.

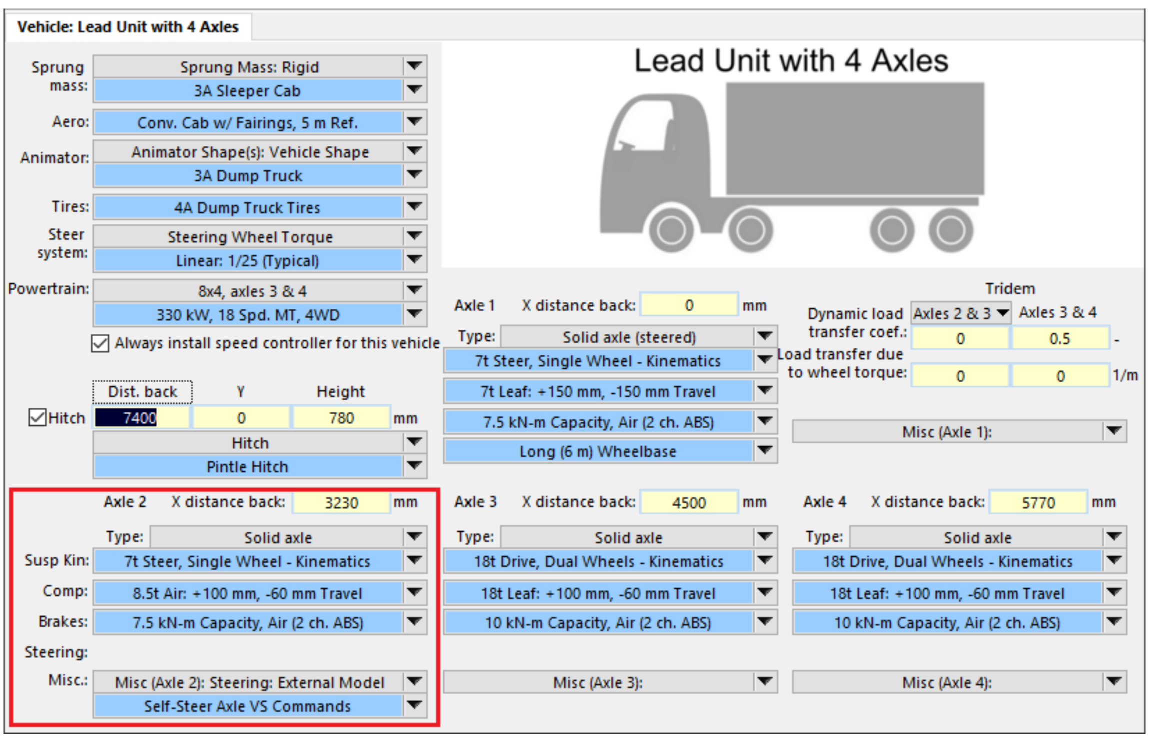

Figure 4. Vehicle assembly screen with self-steer axle datasets highlighted.

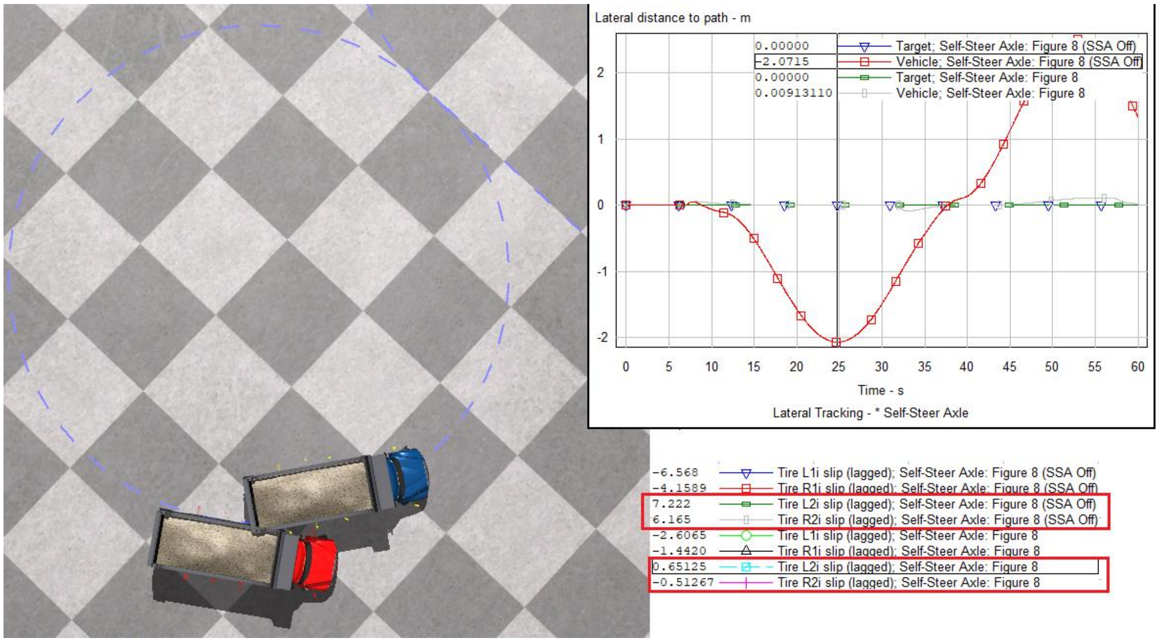

Figure 5. Low speed maneuvering, 12 m radius @ 10 km/h, self-steer axle vs. non-steered axle.

The self-steer axle dump truck is generally well-behaved in the scenarios considered, although there is some slight fluctuation in the steer angles during the 100-0 km/h braking test (Figure 6).

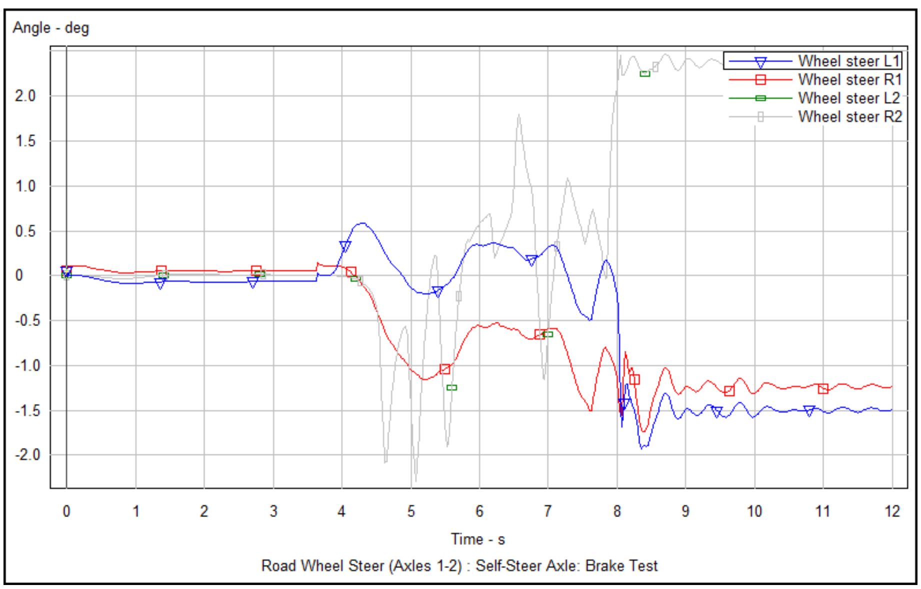

Figure 6. Oscillations in the self-steer axle during the braking test.

The raise/lower example drops the self-steer axle down in the middle of a turn, which is extreme, but the steer angles settle quickly (Figure 7).

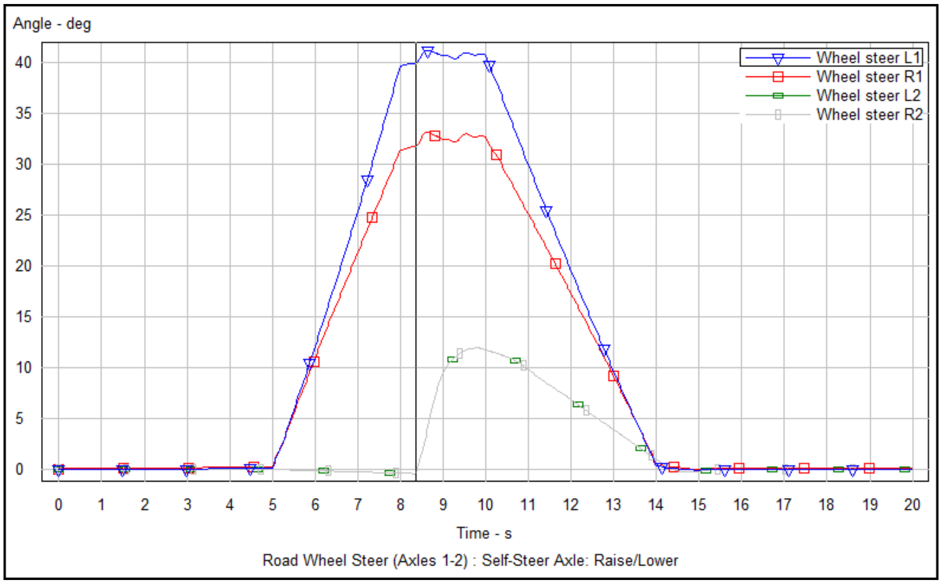

Figure 7. Steer results when the self-steer axle is lowered beginning at T=7 s.

Despite the simplicity of the model used for this example, performance is generally consistent with expectations. The model covered in this example could easily be extended to include other effects such as compliance, friction, steering stops, etc. (Involving nonzero caster or kingpin inclination angle accurately would be more mathematically involved, given kingpin rotation angle is no longer equal to steer angle.)

In terms of tuning for stability, the primary parameter seems to be the steering stabilizer damping c, defined in VS Commands as `SSA_DMP`. The examples use a value of 10 N-s/mm. Another possible extension of the model is to enable/disable the self-steer effect depending on the vehicle's maneuver, simulating a lock-out mechanism. In fact, there is a parameter which does something similar implemented as `SSA_OFF`, which sets the steer of the axle to zero if `SSA_OFF` is not equal to zero. (This was added to compare the same vehicle with and without the self-steer effect.)

Conclusion

This technical memo covered an example of a self-steer axle implemented with an external model and VS Commands. This simple model includes parameters representing the kingpin longitudinal and lateral offsets, steering stabilizer mounting angle and damping, and wheel mass and inertia. This model could be extended to include more complicated effects such as compliance, friction, or even 3D kingpin geometry. On the other hand, the model is stable for the scenarios considered and demonstrates the expected reductions in both turning radius and tire slip angles. Additionally, if needed, additional VS Commands may be used to raise and lower the axle during the simulation.

Aside from doing the external model calculations with VS Commands, one could also consider using embedded Python or Simulink. If nothing else, this example illustrates the extensibility of the TruckSim steering system and provides a template for other custom model extensions.