Tutorial 22

Co-Simulation with CarSim, BikeSim, TruckSim

Objective
This tutorial shows step by step how SimulationX models can be connected to VehicleSim® (CarSim®, BikeSim®, TruckSim®) using co-simulation. VehicleSim is a toolset for the efficient simulation of cars, trucks and motorcycles [1]. SimulationX models and VehicleSim models are connected by using a special co-simulation block in the SimulationX model. This block utilizes the VehicleSim API (Application Programming Interface) [2] for data exchange and synchronization of SimulationX and VehicleSim.
It is assumed that the reader is familiar with the basic functionality both, of SimulationX and VehicleSim. For an introduction to SimulationX, please refer to "Tutorial 1: Getting Started."

The following software modules are needed to execute this tutorial:
  SimulationX incl. a valid license for the option Co-Simulation
  "VehicleSim (CarSim, BikeSim, TruckSim)"
  VehicleSim (CarSim, BikeSim, TruckSim) Version 8

Workflow
The following steps have to be carried out for doing co-simulation between SimulationX and VehicleSim:

  • Extend CarSim with SimulationX models
  • Seamless integration of SimulationX models within CarSim, BikeSim, TruckSim
  • Full flexibility of SimulationX modeling available in CarSim
  • Postprocessing in CarSim and SimulationX

Prepare the VehicleSim model
Prepare the SimulationX model
Run the simulation in SimulationX and VehicleSim
Post processing in VehicleSim and SimulationX

The co-simulation was tested with CarSim. Hence we will use the term CarSim in the following description. It should work with BikeSim and TruckSim in the same way.
The Sample Model

In this tutorial we show how the SimulationX model of an engine combined with a powertrain replaces the engine model of CarSim. The SimulationX model is shown in the following figure.

![SimulationX model of the engine with a very simple dual mass flywheel model.](image)

Figure 1: SimulationX model of the engine with a very simple dual mass flywheel model.

All engine parameters are kept at their default values. The changed parameters of the other model objects are shown in the following table.

- Create this model in SimulationX

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>dms1.J</td>
<td>0.02</td>
<td>kgm²</td>
</tr>
<tr>
<td>dms1.om0</td>
<td>800 (Fixed)</td>
<td>rpm</td>
</tr>
<tr>
<td>dms2.J</td>
<td>0.02</td>
<td>kgm²</td>
</tr>
<tr>
<td>dms2.om0</td>
<td>800 (Fixed)</td>
<td>rpm</td>
</tr>
<tr>
<td>springDamper.kind</td>
<td>Spring-Damper</td>
<td></td>
</tr>
<tr>
<td>springDamper.k</td>
<td>10000</td>
<td>Nm/rad</td>
</tr>
<tr>
<td>springDamper.b</td>
<td>10</td>
<td>Nms/rad</td>
</tr>
<tr>
<td>injection.F</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Preparation of the CarSim Model

At first the CarSim model has to be prepared. We have to change the powertrain configuration to replace the internal engine model by our SimulationX engine.

- Open a database in CarSim.
- Open the powertrain screen in CarSim and switch to "External engine model" as shown in the following figure.
The block diagram changes as shown above. In order to replace the internal engine model the engine speed is to be computed by the SimulationX model and to be fed to CarSim as IMP_AV_ENG. The reaction torque of the driveline EXP_M_EngOut has to be fed back to the powertrain in SimulationX.

To prepare this data exchange the CarSim variable IMP_AV_ENG has to be imported into CarSim from the SimulationX model.

- Change to the "CarSim Run Control" page, click the "Models" drop-down list and change to "Models: Self-Contained Solvers."

Click in the text box below to create a new dataset, call it SimulationX and open it.

On this screen change one of the "Misc. Links" to "I/O Channels/I/O Channels: Import" as shown below.
Figure 4: Prepare the variable import

- Create a new dataset as usual, call it Engine, choose a readme file and add the variable IMP_AV_ENG to the list of “Active Import Variables”.

Figure 5: Add the import variable

- Select “Replace” for mode as shown in the above figure. If the “Mode” selector is disabled, open the associated Parsfile and do the change there.

How this variable is connected to a SimulationX output and how CarSim outputs are connected to SimulationX inputs is shown in the next section.

Preparation of the SimulationX Model

Our engine model has to be changed a little bit. To feed the torque EXP_M_EngOut computed in CarSim back to the SimulationX model we add an External Torque (Library: “Mechanics → Rotation → Source”) element.

- Add the element to the model as shown in Figure 6 and rename it to loadTorque.
- Leave the parameter T at its default setting “in1”. In this way the value of the signal input “in1” is fed as load torque to the driveline. If you want to connect your other variables to CarSim take care of the sign.
- The powertrain speed will be sent to CarSim and assigned to the variable IMP_AV_ENG. Hence we add a speed sensor (Library: “Mechanics → Rotation → Sensor”) to our model.
Figure 6: Engine model with loadTorque and speed sensor

- Now we add the "VehicleSim Co-Simulation" block. It is included in the package on the Library: "CoSimulation → VehicleSim".

Figure 7: SimulationX "VehicleSim Co-Simulation" block

The block has the following properties:

Figure 8: Property window of the "VehicleSim Co-Simulation" block.

Simfile Name (simfile):
To communicate with CarSim using its API, the recently used simfile.sim must be given. This file contains links to the other data files used by CarSim. The file is located in the current database folder. This folder is selected during CarSim start up in the following dialog box.

![Database selection during CarSim start up.](image)

- Press the button at the end of the edit box of the parameter `simfile`. A file selection window opens. Go to the current database folder and select the current CarSim simfile. If you have not run the CarSim model before, there is no file in this folder.

**Step Size ts:**
This step size is used to communicate with CarSim and to update inputs and outputs. The CarSim step size should have the same value. If the CarSim step size is different, an error message is shown and the simulation run does not start.
As stated in [2] chapter "4. Extending Math Models Using Import and Export Arrays") a one step time delay is introduced between CarSim and SimulationX. This may lead to numerical instabilities for closed loop models with a tight coupling. Reduce the step size $t_s$ in such cases.
- For our sample model we use a step size of $t_s = 1 \text{ ms}$.

**Number of Inputs $nu$, Number of Outputs $ny$:**
The block communicates via the signal input $u$ and the signal output $y$ with the enclosing SimulationX model. These signal connectors can carry vectors of variables. The dimension of these vectors is defined by $nu$ (for input $u$) und $ny$ (for output $y$).
For connecting scalar signal connectors to these vector connectors, the multiplexers and demultiplexers of the package Library: “CoSimulation → MuxDemux” can be used (see below).
- In our example one signal (the powertrain speed) is sent to CarSim. Hence we set $nu = 1$.
- The reaction torque of the CarSim model is fed back to the powertrain model. Additionally we want to observe the speed of the car computed in CarSim. Thus set $ny = 2$.

**Import variable $uImp[nu]$, Export variables $yExp[ny]$:**
These parameters are vectors of strings. The dimension is respectively $nu$ or $ny$. Here we must enter the names of the CarSim variables which have to be connected with the block inputs and outputs. If one of the names is not found in the CarSim database, a warning message is shown during start-up of the simulation run.
- The powertrain speed is imported to CarSim (see Figure 5) as `IMP_AV_ENG`. Thus enter (“IMP_AV_ENG”) for $uImp$.
- In addition to the CarSim reaction torque `EXP_M_EngOut` we want to observe the vehicle speed $VX$. Hence enter (“EXP_M_EngOut”,”VX”) for $yExp$.

At the end of the described modifications the block should have the following properties:
As last step the co-simulation block is to be connected with the powertrain model. We connect the input $u$ with the output of the speed sensor ($speed.om$).

The output has the dimension $ny=2$. Hence we need a demultiplexer (Library: "CoSimulation → MuxDemux → Demultiplex2") to split this vector into scalars. In this way we end up with the following model.

Now the SimulationX model is ready for co-simulation with CarSim.

If you want to transfer more than one signal to CarSim you have to adapt the dimension by using the parameter $nu$ and use the corresponding multiplexer block to combine the signals to a vector.

In the package Library: "CoSimulation → MuxDemux" multiplexers and demultiplexers of dimension 2 to 5 are provided. If you need higher dimensions you can easily create your own blocks using the existing ones as guideline. The implementation of these blocks is open and can be viewed by using the SimulationX TypeDesigner (see Tutorial 5: TypeDesigner).

Co-Simulation

After the following settings on the CarSim "Self Contained Solvers" page, the co-simulation can be started. Change now to CarSim and open the dataset we created at the beginning by clicking on the SimulationX button on the Run Control screen.
• Switch on "Use external wrapper program" and select the executable "VehicleSimXStarter.exe" as "External wrapper program". The "VehicleSimXStarter" application depends on the used SimulationX version and the path to the right depends on the operating system version. It is located in the subdirectory "ITI-Software/SimulationX/Tools" of the following folder:
  o SimulationX 32bit on Windows 32bit "program files" folder
  o SimulationX 32bit on Windows 64bit "program files (x86)" folder
  o SimulationX 64bit on Windows 64bit "program files" folder

• On 64bit operating systems the "For 64-bit Windows OS" drop down box is displayed. Choose here the same solver version as the used SimulationX version, 32bit or 64bit. Make sure, the "Specify alternative VS solver DLL files" checkbox is unchecked, if no solver is selected.

Figure 13: Select the VehicleSimXStarter.exe as external wrapper program.

• Furthermore input 0.001 s as Time step. This value must be equal to the parameter $t_s$ of the SimulationX co-simulation block.

Now the simulation run can be started by pressing the "Run Math Model" button on the CarSim main page. Make sure that SimulationX is running and the model containing the "VehicleSim Co-Simulation" block is the active one. Instead of the CarSim internal computation routine the VehicleSimXStarter application is started. This application executes the following actions:

• Connects itself to the running SimulationX instance, or opens a new SimulationX instance.
• Connects itself to the active model.
• Checks whether this model contains a "VehicleSim Co-Simulation" block.
• Resets the active model if necessary.
• Starts the simulation in SimulationX. Now the "VehicleSim Co-Simulation" block communicates via the VehicleSim API with CarSim.
• Shows status messages during the simulation run.
• Closes itself after the simulation run.
• If SimulationX could not be found or opened, or if the active model does not contain a "VehicleSim Co-Simulation" block an error message is shown and the simulation run is not started.

Figure 14: The VehicleSimXStarter is connected to SimulationX, the co-simulation is running.

The co-simulation run stops under the following conditions:
• If the Stop button of the VehicleSimXStarter is pressed. It can be continued by pressing the Start button.
• If the CarSim stop time is reached or a CarSim stop condition becomes true. In this case a message in SimulationX is shown.
• If the Stop button in SimulationX is pressed, the stop time is reached, or a termination condition becomes true. The VehicleSimXStarter closes automatically in that case. The simulation can be continued by pressing the SimulationX Start button.

The simulation run can be started as usual from SimulationX too. In this case it is possible that the CarSim model data are not transferred to the database files which are loaded by the CarSim API during co-simulation. Even if you save the CarSim database the data are not written to these files. This is done by CarSim, when you start the external wrapper program, the VehicleSimXStarter in our case. Hence it is safer to start the co-simulation from CarSim. It is safe to start the co-simulation in SimulationX if you already started it once by using CarSim and did not change the CarSim model meanwhile.

**Post processing in CarSim and SimulationX**

After co-simulation runs the usual post processing functions of CarSim (Animate, Plot) and SimulationX are available. The following figure shows the SimulationX result window and the CarSim plot of the vehicle speed.

![Vehicle speed in SimulationX and CarSim.](image)

**Common Pitfalls**

Please take care on the following points to avoid problems:

• The "VehicleSim Co-Simulation" block is contained in the package CoSimulation. This package is stored by the installer in the first entry of the SimulationX Modelica Search path (Menu: Extras/Options/Directories). If you change or delete this path later, the package cannot be found.
• The "VehicleSim Co-Simulation" block uses external functions implemented in the DLL ITI_CSVehicleSim.DLL. This file is stored by the installer in the first entry of the SimulationX External Functions path (Menu: Extras/Options/Directories). If you change or delete this path later, the external function cannot be found and the co-simulation will not start.
• There are two versions of the "VehicleSimXStarter.exe", one for the 32bit and the 64bit SimulationX. If the selected "VehicleSimXStarter.exe" did not fit the used SimulationX version, you will get a message box, followed by an additional empty SimulationX instance. Make sure that the right "VehicleSimXStarter.exe" is selected.
• While using the SimulationX 64bit version, it is important to use also the 64bit version of the CarSim solvers.
• The SimulationX model is computed beside the CarSim model. Before each CarSim time step:
  - the inputs for the SimulationX models are read,
  - the SimulationX model time step is computed,
the outputs which are imported to the CarSim model are updated.

In this way a one time step delay is introduced for closed loop models (as stated in [2] chapter "4. Extending Math Models Using Import and Export Arrays"). This may lead to numerical instabilities for models with a tight coupling. Reduce the CarSim step size in such cases.

Comparison of Co-Simulation and SimulationX Code Export

Beside of this co-simulation interface you can connect SimulationX models with VehicleSim using the code export target “CarSim, BikeSim, TruckSim component” (see Tutorial 20: “Model Export to CarSim, BikeSim, TruckSim”). The following hints should help to decide which option is best suited to you:

- Using the code export target the SimulationX model is exported to C-code, compiled and integrated with VehicleSim. These steps have to be carried out every time the SimulationX model is changed. In a design stage with frequent model changes this takes a little bit time.
- In the co-simulation case SimulationX and VehicleSim run in parallel. Thus model changes in SimulationX can be done as usual. There is no time overhead due to code export and compilation.
- Beside of this, post processing in SimulationX can be done as usual doing co-simulation. You do have access to all model parameters and variables as usual. Using the code export option it is only possible to access the SimulationX variables which are defined as outputs for post processing.
- Once code export is done, the exported SimulationX model runs without needing a SimulationX license. This eases the model exchange with partners and customers.
- After code export the SimulationX model functionality cannot be seen by the user. The model algorithm is compiled and thus hidden. This safes your intellectual property (controller algorithms, parameters of technical components). Additionally the exported model cannot be changed by the user. Both points are important for model exchange with partners and customers.

Literature

[1] www.carsim.com