Tutorial 20

Model Export to CarSim, BikeSim, TruckSim

Objective
This tutorial shows step by step how SimulationX models can be exported to VehicleSim® (CarSim®, BikeSim®, TruckSim®). VehicleSim is a toolset for the efficient simulation of cars, trucks and motorcycles [1]. SimulationX models can be exported using the Code Export wizard. The VehicleSim API (Application Programming Interface) [2] is used for the seamless integration of SimulationX models within 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 Code Export Target "VehicleSim (CarSim, BikeSim, TruckSim)"
- VehicleSim (CarSim, BikeSim, TruckSim) Version 8
- Microsoft Compiler (e.g. the free Microsoft Developer Studio Express Edition)

Workflow
The following steps have to be carried out:

- Prepare the VehicleSim model
- Prepare the SimulationX model
- Code Export of the SimulationX model
- Integration and simulation in VehicleSim
- Post processing in VehicleSim

The model export 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 is exported to CarSim. The model is shown in the following figure.

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.
Figure 2: Change to External engine model

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 engine in SimulationX.

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

Figure 3: Switch to 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.
• 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 4: Prepare the variable import](image)

• Select "Replace" for mode as shown in Figure 5. 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 below.

**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 the following figure and rename it to **loadTorque**.
Figure 6: Engine model prepared for export to CarSim

- 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 export your own SimulationX models take care of the sign of the exchanged variables.

SimulationX models exported to CarSim contain a fixed step solver. Before you carry out the code export you should test whether your model can be computed by a fixed solver and which step size should be used. The SimulationX "Fixed Step Solver" can be selected via the menu "Simulation → Transient Settings... → General". The "Min. Calculation Step Size" $dtMin$ is used as integration step size for the Fixed Step Solver inside SimulationX and as the default value for the model exported to CarSim. This step size can be changed in CarSim.

- Our tutorial model runs stable with a step size of $dtMin = 5e-4$ s. Set this value in SimulationX.

**Code Export of the SimulationX Model**

Now the model can be exported to CarSim.

- Open the Code Export Wizard using the menu "Export → C-Code...".
- At first select "CarSim, BikSim, TruckSim component" as Project Type, enter a Project Name and select a Project Path as shown below.

![SimulationX Code Export Wizard](image)

- On the next wizard page you have to select the inputs for the SimulationX model which can be connected to CarSim variables. In our sample we have to feed the torque $EXP_M_EngOut$ computed by CarSim as load torque for our engine model. Hence select the input $loadTorque.in1$ and enter $EXP_M_EngOut$ as "VehicleSim Name".
Inputs, which are not assigned to a CarSim variable, are set to zero during the simulation run.

Next we have to select the outputs. The speed on the right side of our model is to be connected to the CarSim import variable **IMP_AV_ENG**.

- Select `dms2.om` as output and enter `IMP_AV_ENG` as "VehicleSim Name".

Model variables which shall only be observed in CarSim can also be selected. If no "VehicleSim Name" is entered the variable is added to the CarSim database using its SimulationX identifier. CarSim supports strings with a maximum length of 8 characters using ALL CAPS (uppercase). Hence the SimulationX identifier is truncated and transformed to ALL CAPS before adding it to the CarSim database. If a given "VehicleSim Name" cannot be found in the CarSim database the variable is added using this name.

- Add the output `engine.Te` without the "VehicleSim Name"
CarSim uses SI base units for its computation in the same way as SimulationX does. Hence you do not have to take care for unit conversions at all.

On the next page we can select parameters which shall be changeable in CarSim. Once again a "VehicleSim Name" can be entered. Below we will show how these parameters can be changed in CarSim. The parameters keep their original values from SimulationX if you do not change them in CarSim.

- We select the amount of injection \( (\text{injection.F}) \) and some engine parameters \( (\text{engine.Pn}, \text{engine.omn}) \).

![Figure 10: Selection of parameters](image)

When you change to the next Code Export Wizard page the model is analysed and the C-code is generated. As last step we have to compile the generated code.

- Change to the "Post-processing" page and press the "Build" button to do this. One of the supported Microsoft compilers has to be installed on your computer. If you want to compile for 64bit systems, you have to tick the "Compile for x64 Platform" checkbox.
Figure 11: Compilation of the generated code.

The result of the compilation is an executable program. It contains all necessary calls to the CarSim API functions to connect the inputs and outputs and to carry out the necessary computations.

**Integration and simulation in CarSim**

To integrate the generated executable within CarSim you have to select it as external wrapper program.

- Open the dataset we created at the beginning by clicking on the "SimulationX" button on the "Run Control" screen.

- Select "Use external wrapper program" and select the executable generated by SimulationX as shown below. Make sure, the "Specify alternative VS solver DLL files" checkbox is unchecked, if no solver is selected. If you have used the "Compile for x64 Platform" option during the code-export, select "Use 64-bit Solvers" in the "For 64-bit Windows OS:" drop down box.

Figure 12: The SimulationX dataset
Figure 13: Select the generated executable as external wrapper program

- The computation starts as usual by clicking on "Run Math Model" on the main page.

Before the computation starts some information about changed parameters, connected inputs/outputs and the progress are shown in a console window. This window is closed by default after the simulation run.

- To keep it open enter OPT_PAUSE 1 under "Miscellaneous settings for advanced users" (Figure 14).
- To change parameters enter their name followed by the new value (in SI units) to the command edit field as shown below.

Figure 14: Complete dataset for integration of the SimulationX model within CarSim.
The solver parameters of the SimulationX model can be changed here too. Our generated executable adds parameters for the integration step size \( SIMX_DT \) and the integration method \( SIMX_MODE \) as parameters to the CarSim database. They can be changed as shown above.

The SimulationX integration step size should be smaller or equal to the CarSim one. The CarSim step size should be an integer multiple of the SimulationX integration step size \( SIMX_DT \). In this way the SimulationX model can be computed by using a step size smaller than the one used by the CarSim model.

The parameter \( SIMX_MODE \) changes the used integration method as follows:

<table>
<thead>
<tr>
<th>( SIMX_MODE )</th>
<th>Integration Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Euler Forward</td>
</tr>
<tr>
<td>1</td>
<td>ITI Standard</td>
</tr>
<tr>
<td>2</td>
<td>Heun’s Method</td>
</tr>
<tr>
<td>3</td>
<td>RKF23</td>
</tr>
<tr>
<td>4</td>
<td>DOPRI5</td>
</tr>
</tbody>
</table>

- Running the simulation after these changings leads to the following outputs during the simulation run.

![Figure 15: Outputs during the simulation run](image)

The parameters, inputs, and outputs added to the database or assigned to CarSim are listed. Trace, warning, and error messages of the SimulationX model and statistics are shown.
Post processing in CarSim

After the simulation run, the normal post processing functions (Animate, Plot) are available. Beside CarSim variables also the outputs of the SimulationX model can be plotted. You can select these outputs as usual in the plot tool. The data in the following figure may differ according to the car configuration. The CarSim output file step size was set to 0.001 s to increase the resolution of the plotted data.

Figure 16: Selection of the engine torque output and zoomed plot window

Comparison of SimulationX Code Export and Co-Simulation

Beside of this code export target you can connect SimulationX models with CarSim using the co-simulation interface (see Tutorial 22: “Co-Simulation with 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 CarSim. 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 of time.
- In the co-simulation case SimulationX and CarSim 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 safe 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.
Common Pitfalls

Please take care of the following points to avoid problems:

- If you do not enter a "VehicleSim Name" for outputs and parameters, the variables are added to the database by using truncated SimulationX identifiers. Make sure that these truncated identifiers are unique.
- The exported SimulationX model contains a fixed step solver. You should test inside SimulationX if your model can be computed stably with a fixed step solver.
- 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 for the inputs and outputs occurs 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.

Literature

[1]  www.carsim.com