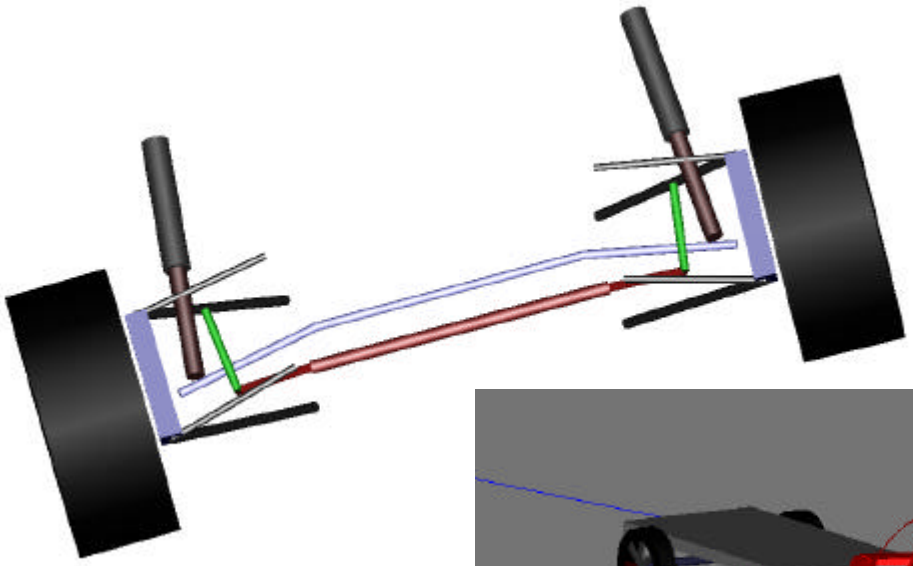
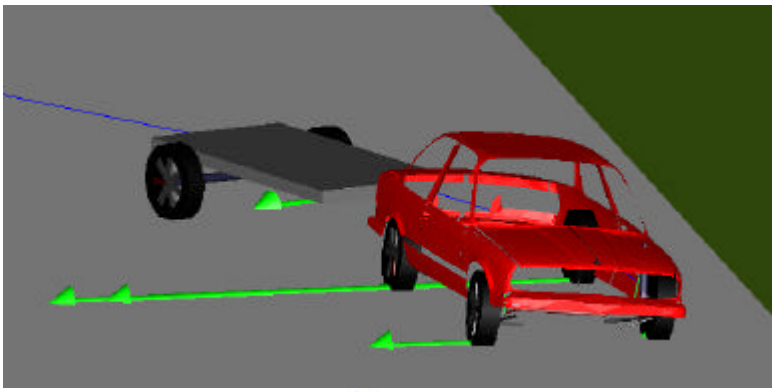
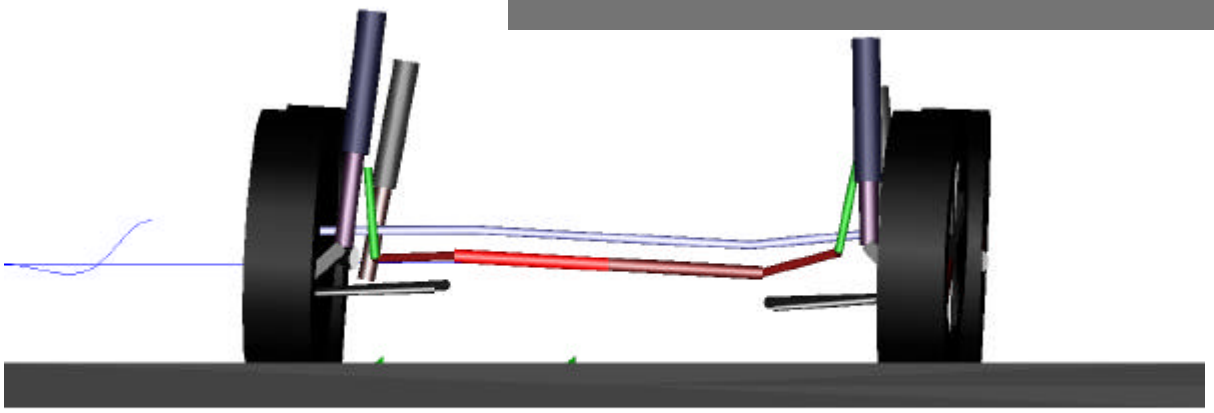
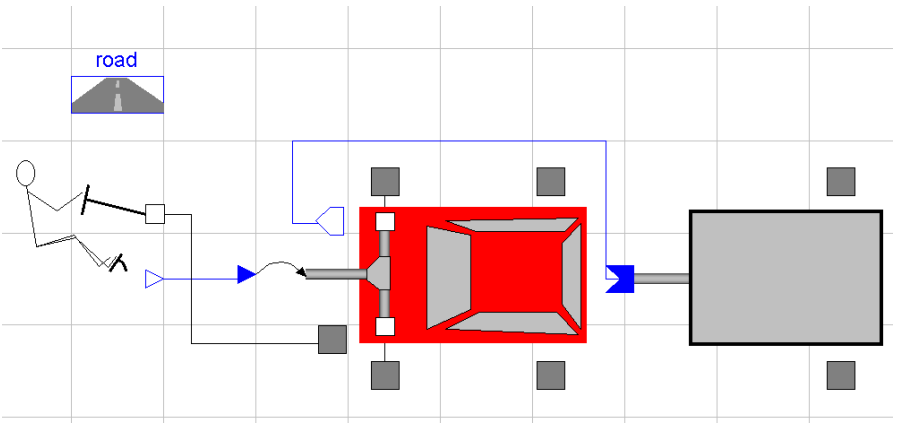
Getting started with VehicleDynamics.mo



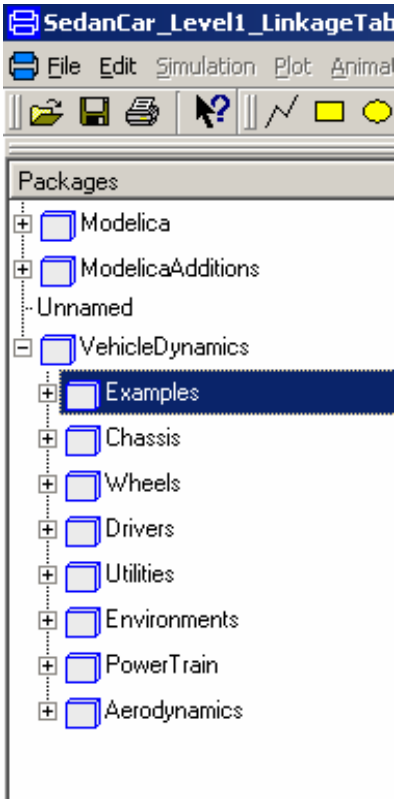






Introduction

This guide will take you through some examples in order to get you started using the Chassis library in VehicleDynamics. The guide will show you, how to build a Vehicle model and how the model can be adjusted for your needs. Before you do this, you are recommended to have worked through the 'Getting started with Dymola'. To learn more about defining own components, check the `VehicleDynamics.Chassis` documentation. Before you continue, make sure that your copy of the `VehicleDynamics.mo` is opened. To check this, make sure you have a box named VehicleDynamics with in the package window.



Building a VehicleDynamics model

Start a new model by selecting File/New/Model and call it `MyCar`. This will now appear in the package window at the bottom of the listing. If the model is not selected, select it by double-clicking it.

First of all, we need a chassis model. Chose the `VehicleDynamics.Chassis.StandardChassis` model and drag it to the diagram layer of the model `MyCar`. Right click the model, select `Parameters` and change the Name to `chassis`. If it appears small on the screen, select it and increase it by dragging one of the read squares appearing at two corners of the model. You should now see an icon of a chassis with six connectors. These correspond to the wheel axes, the steering wheel axis and a contact point at the Body Geometric Reference point (`BGR`) at the body. Soon, we shall se how these can be used to add other components but first we shall add a road to the model.

The road supplies the chassis with information about the shape and surface conditions, without this the model will not work. The reason that this is not included in the chassis model itself is that it should be easy to change from one road to another. Drag the `VehicleDynamics.Environments.SplitMueRoad` to your model and rename it to `Road`. If you swap to the Modelica Text layer, you will see:

```
VehicleDynamics.Environments.SplitMueRoad Road annotation(…);
```

Replace the row that declares the road with

```
inner block Road = VehicleDynamics.Environments.SplitMueRoadBlock;
```

In this case it is necessary to write exactly `Road` since the chassis model is looking for a road to find the right information.

Now you have a model that can be simulated, but before we try this out we will add some more components to the model.

To be able to control the chassis' motion, you need a driver. Choose the `VehicleDynamics.Drivers.OpenLoopDriver`. Change the name to `driver` and connect it to the rotational flange at the front of your chassis model icon. If you double click the line you should now see a dialog window `connect(driver.flange_SW, chassis.flange_SW)`, i.e. the driver is connected to the steering wheel of the chassis. Note that the order of how the names within the parenthesis occur makes no difference. Additional information like `annotation…` only concerns the visual appearance of the connecting line.

As a next step, you will add a simple power train. If you have been working with the `PowerTrain` or `SimpleCar` libraries earlier, you can add your own models in the same way as presented here, all libraries are compatible with another. Here we will apply a very simple model of a step torque input at the front wheels to show how power trains and brakes can be applied.

Add the `VehicleDynamics.Utilities.SimplePowerTrain` to the model and change the name to `powerTrain`. Connect the right and left flanges of `powerTrain` to the left and right front wheels of `chassis`. If you click the connections the should now say `connect(powerTrain.flange_1, chassis.flange_1)` and `connect(powerTrain.flange_2, chassis.flange_2)`, respectively. Of course, you can just as well add `powerTrain` to the rear wheel or apply another model to all wheels. Note that there is no restriction to how many connections you can have on one wheel flange. Also connect the driver with the `powerTrain`. The next step is to translate and simulate the model.

Translate and Simulate

Before the model can be translated and simulated, some settings have to be adjusted. To do this, swap to 'Simulation mode' and run the `TranslationSettingsChassis.mos` that is found in the Script-folder that comes along with the `VehicleDynamics.mo` library. The settings within the script allow Dymola to handle the state selection in a clever way.

Now it is time to translate your model. (If you have a Visual C++ compiler, it is recommended to use this. In many cases the GCC default compiler fails.) Swap back to the

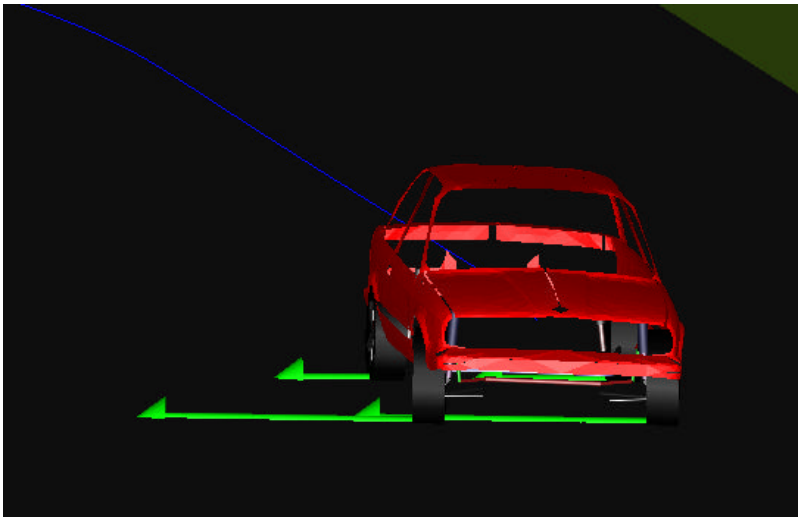
‘Modeling mode’ and make sure that you have selected the `MyCar` model. Return to the ‘Simulation mode’ and start the translation by pressing the translate button or choosing Simulation/Translate. You will now get a warning saying:

```
…the variable driver.position.phi has been deselected as a continuous time
state…
```

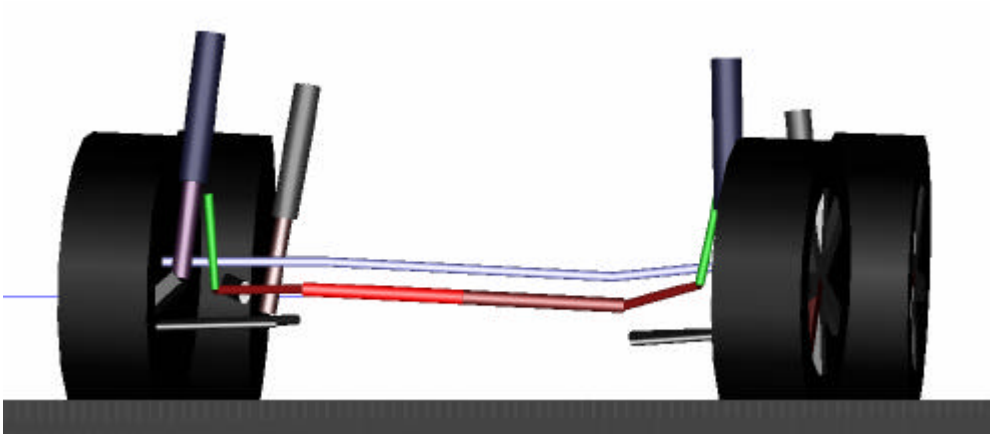
This is because the driver model used imposes a defined motion on the steering wheel and not a torque that a more detailed driver model would do. Thus you can ignore this warning.

Now you can simulate the model. Set the simulation time to 7s. This takes about 20s to simulate on a PIII-800MHz notebook with 256MB ram.

If no animation window is open, select ‘Animation/New Animation Window’ to open one. You will now see the front of the car from the top. To get a better view of the car, set a new iewpoint with the 3D View Control (‘Animation/3D View Control’). Also make sure to select Follow selected object in the Animation setup dialog box (‘Animation/Setup’). Click on the animation window to select which part you want to let the animation follow. Now, run the animation and you’ll see you model performing a double lane change manoeuvre. The tyre forces can be visualised as shown below by selecting ‘Animation/Setup…/Vectors’ and setting ‘Forces’ unequal to 0, appropriate values are typically around 0.01-0.001 m/N.



If you want to see the animation with out body graphics, the easiest way is to rename/hide the body graphics file (by default ‘2.dxf’) so that it cannot be found by Dymola. There is no need to resimulate but it might be necessary to reopen the result file (‘Animation/Open result file…’). It is by default named as the model simulated, in this case ‘MyCar.mat’.



Handling parameters

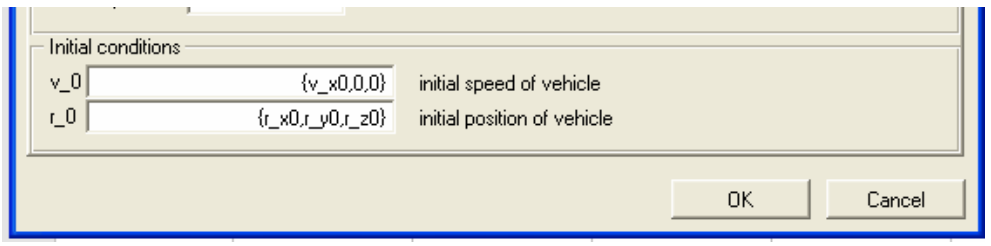
You might have noticed that there are no parameters that you can change in the model you just simulated, although the models you have used contain parameters. This is because all parameters are evaluated in order to make the model more efficient. Though, it is often interesting to study the effects of variation of a set of parameters and you will now add flexibility to the model by preventing some parameters from being evaluated.

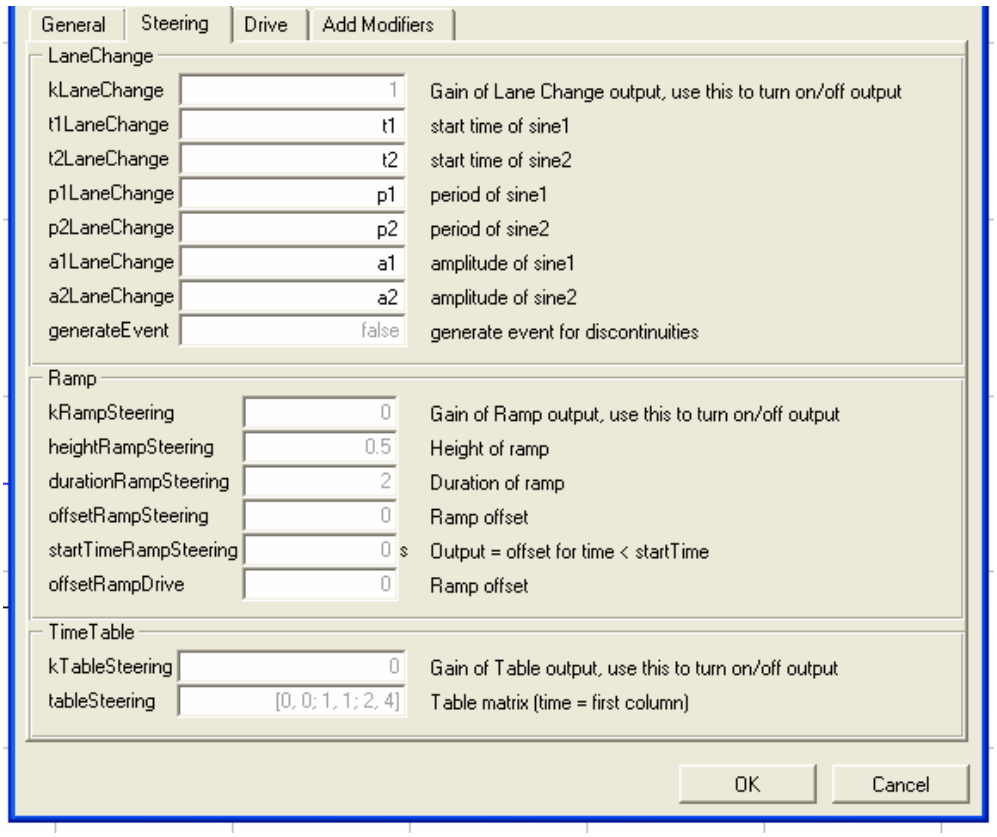
Currently, the only way to do this is to propagate the interesting parameters to the top model. In this case you will select parameters to control the initial conditions of the VehicleDynamics, the location of the body's centre of gravity and the steer input form the driver. Add the following to your car model:

```
// INITIAL CONDITIONS
parameter Real v_x0=19 "|Initial conditions| start velocity x";
parameter Real r_x0=-120 "|Initial conditions| start position x";
parameter Real r_y0=-2.5 "|Initial conditions| start position y";
parameter Real r_z0=0.28 "|Initial conditions| start position z";

//LANE CHANGE
parameter Real t1=0.5 "|Steering|LaneChange| start time of sine1 ";
parameter Real t2=3.5 "|Steering|LaneChange|start time of sine2";
parameter Real p1=2 "|Steering|LaneChange|period of sine1";
parameter Real p2=2 "|Steering|LaneChange| period of sine2";
parameter Real a1=0.5 "|Steering|LaneChange| amplitude of sine1";
parameter Real a2=-0.5 "|Steering|LaneChange| amplitude of sine2";
```

To complete the propagation, these parameters must be related to those within the model. Define the following within the chassis and driver components by double-clicking them:

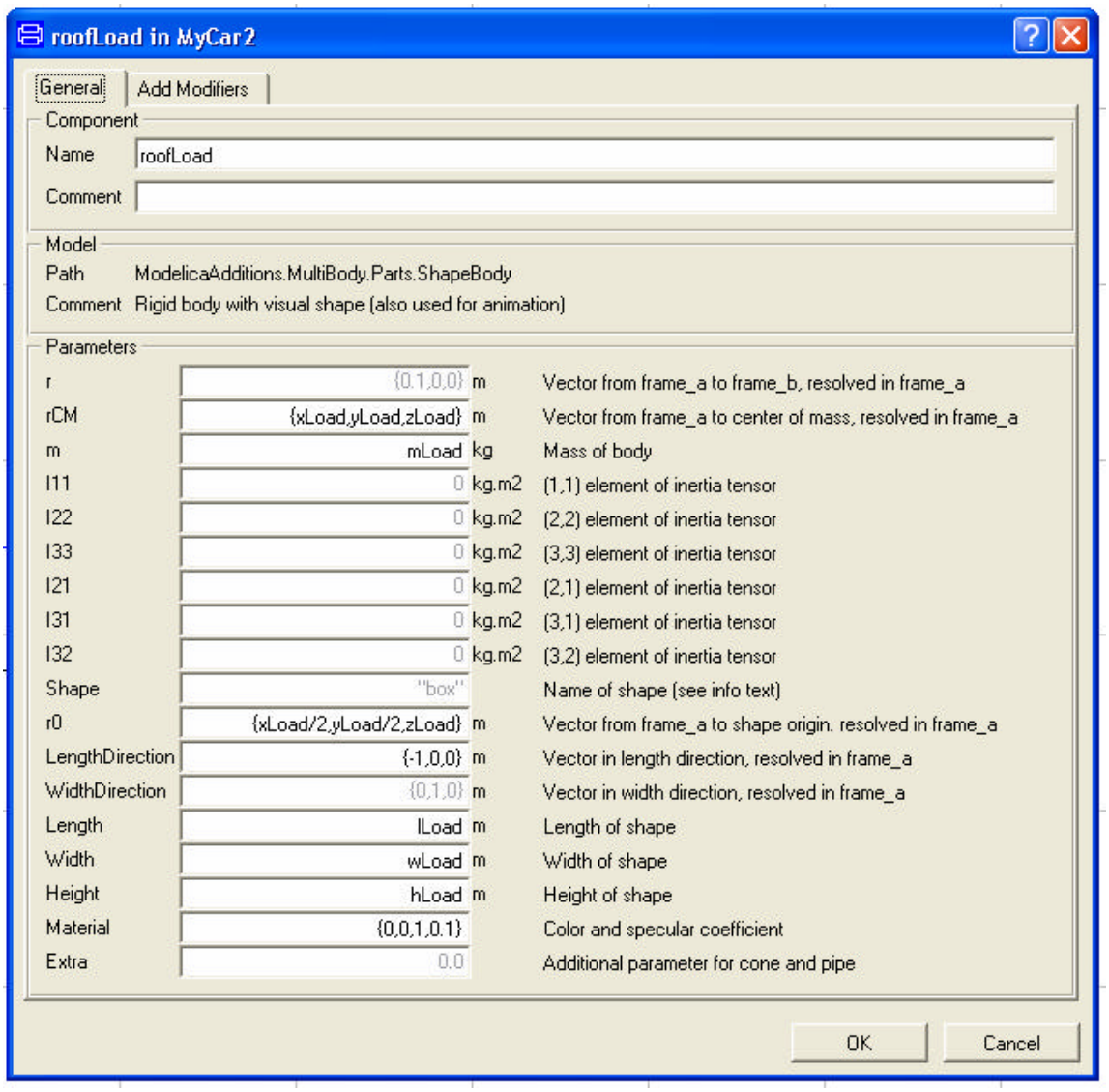




Additional components

An additional component shall be added to demonstrate how to use the BGR connector. In this case you will add an additional load at the roof but you may just as well add trailers, aerodynamic models and much more.

Add a `ModelicaAdditions.MultiBody.Parts.ShapeBody` to the model, rename it to `roofLoad` and connect it to the `BGR` connector at the chassis component. Propagate the location and the mass properties of the load like this




```
// INITIAL CONDITIONS
parameter Real v_x0=19 "|Initial conditions| start velocity x";
parameter Real r_x0=-120 "|Initial conditions| start position x";
parameter Real r_y0=-2.5 "|Initial conditions| start position y";
parameter Real r_z0=0.28 "|Initial conditions| start position z";

//LANE CHANGE
parameter Real t1=0.5 "|Steering|LaneChange| start time of sinel ";
parameter Real t2=3.5 "|Steering|LaneChange|start time of sine2";
parameter Real p1=2 "|Steering|LaneChange|period of sinel";
parameter Real p2=2 "|Steering|LaneChange| period of sine2";
parameter Real a1=0.5 "|Steering|LaneChange| amplitude of sinel";
parameter Real a2=-0.5 "|Steering|LaneChange| amplitude of sine2";

// ADDITIONAL LOAD
parameter Real xLoad=-1.6 "|Additional load| x displacement of additional load, relative the cars BGR";
parameter Real yLoad=0 "|Additional load| y displacement of additional load, relative the cars BGR";
parameter Real zLoad=1.3 "|Additional load| z displacement of additional load, relative the cars BGR";
parameter Real mLoad=0 "|Additional load| mass of additional load";
parameter Real lLoad=0.5 "|Additional load| length of additional shape";
parameter Real wLoad=0.5 "|Additional load| width of additional shape";
parameter Real hLoad=0.1 "|Additional load| height of additional shape";
```

This set of parameters can be changed without recompilation and variation of could easily be done with scripting.