Tutorial 2

FMI for Composite Modelling, Co-Simulation and Model Exchange

Andreas Heuermann and Lennart Ochel

15th MODPROD Workshop, February 3-4, 2021

LINKÖPING UNIVERSITY

Outline

- Preparation
  - Installation instructions
- Introduction
  - FMI and SSP standards
  - OMSimulator
  - Brief demo
- Exercises / Your examples!
- Wrap-up / Questions

Installation Instructions

Preparation

Installation Instructions

What you will need for this tutorial:

- **OpenModelica** >=v1.17.0-dev installed
- **Python3** installed with modules
  - OMPython
  - OMSimulator (version >=2.1.1)
- **Jupyter Notebook** for Python3

Note for Mac users: Use a Virtual Machine with Linux

Installation Instructions

- Documentation
  - OpenModelica User’s Guide
    openmodelica.org/doc/OpenModelicaUsersGuide/latest/
  - OMSimulator User’s Guide
    openmodelica.org/doc/OMSimulator/master/html/
- Tickets (feature request & bug report)
  - Trac trac.openmodelica.org/OpenModelica/
  - GitHub github.com/OpenModelica/OMSimulator/
- Community
  - OpenModelica Forum openmodelica.org/forum
  - Stack Overflow stackoverflow.com/
  - Discord Modelica chatroom

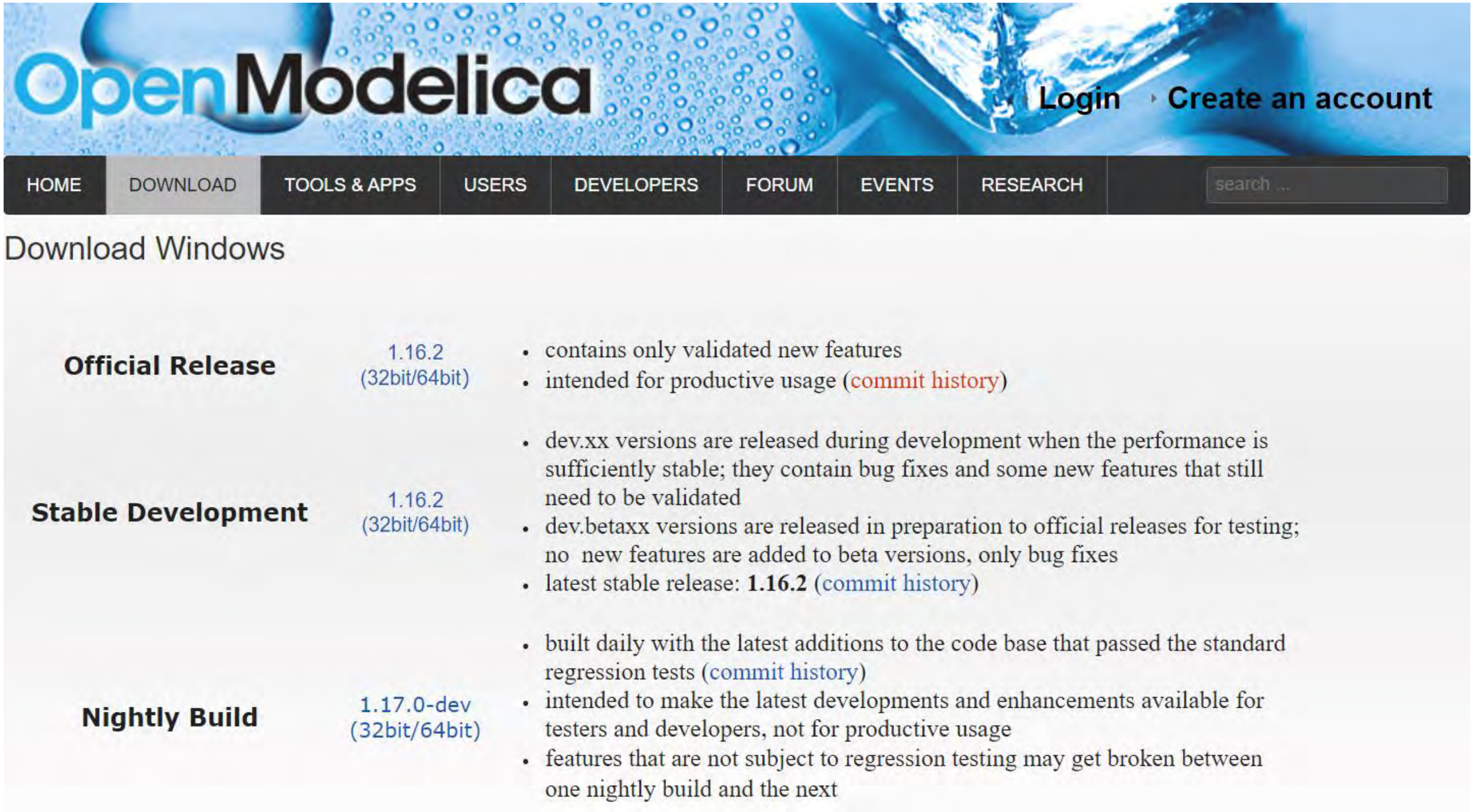
Installation Instructions

- OpenModelica >=v1.17.0-dev
  - OMSimulator is part of **OMEdit**
  - GUI + CLI + scripting available
  - Follow instructions for your platform

openmodelica.org

OpenModelica

Installation Instructions



Download Windows

Official Release	1.16.2 (32bit/64bit)	• contains only validated new features • intended for productive usage (commit history)
Stable Development	1.16.2 (32bit/64bit)	• dev.xx versions are released during development when the performance is sufficiently stable; they contain bug fixes and some new features that still need to be validated • dev.betaxx versions are released in preparation to official releases for testing; no new features are added to beta versions, only bug fixes • latest stable release: **1.16.2** (commit history)
Nightly Build	1.17.0-dev (32bit/64bit)	• built daily with the latest additions to the code base that passed the standard regression tests (commit history) • intended to make the latest developments and enhancements available for testers and developers, not for productive usage • features that are not subject to regression testing may get broken between one nightly build and the next

Installation Instructions

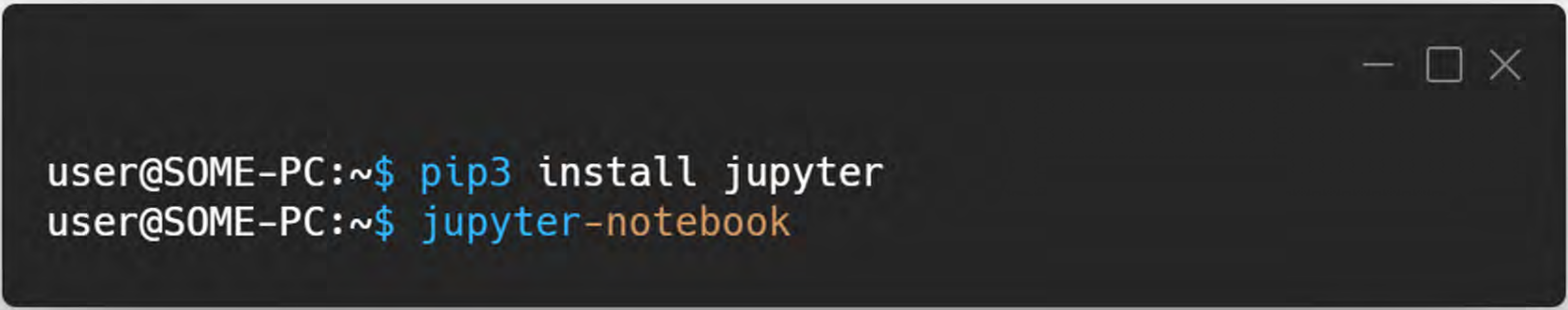
Install **Jupyter Notebook** on Windows:

- Install Anaconda
  - Download latest Anaconda with Python 3.8
    https://www.anaconda.com/
  - Install Anaconda by following its instructions
- Start Jupyter Notebook
  - Windows: Press Win-Key and type “Jupyter Notebook (Anaconda3)“ and launch the app

Installation Instructions

Install **Jupyter** Notebook on Linux:

- Install Python 3.8 and pip3
- Install Jupyter Notebook



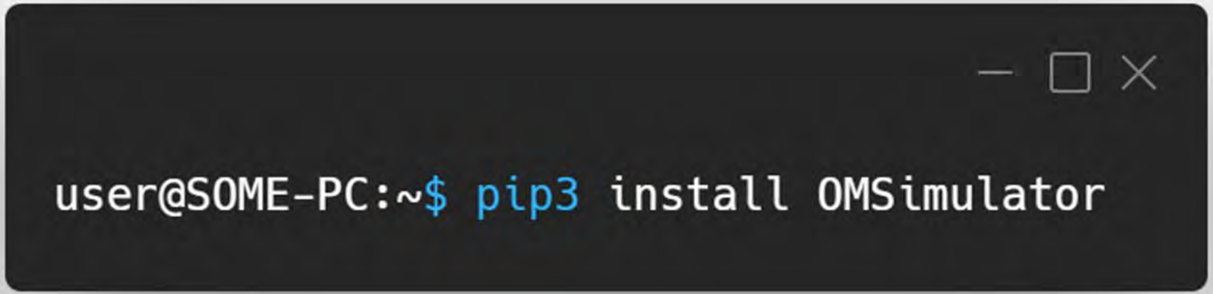
```
user@SOME-PC:~$ pip3 install jupyter
user@SOME-PC:~$ jupyter-notebook
```

Installation Instructions

Install **OMSimulator** (version >= 2.1.1) with pip

- Open a shell with Python in your path
  - Windows: Run app Anaconda Prompt (Anaconda3)

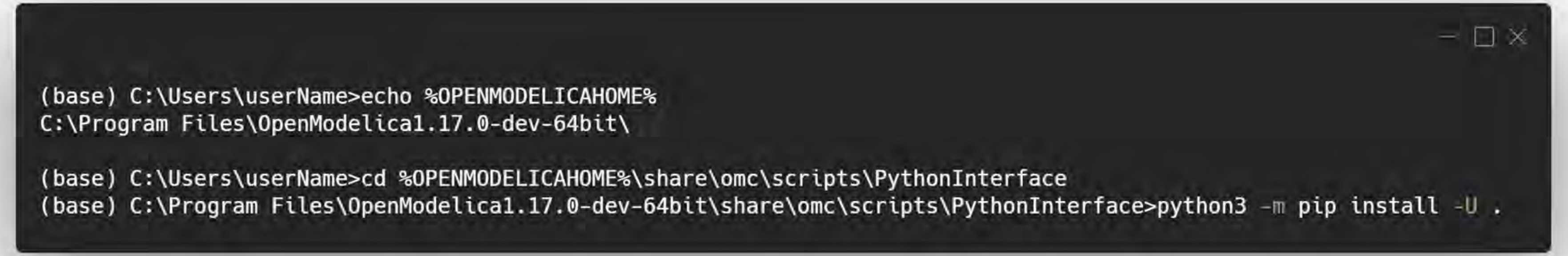
pip3 install OMSimulator



```
user@SOME-PC:~$ pip3 install OMSimulator
```

Installation Instructions

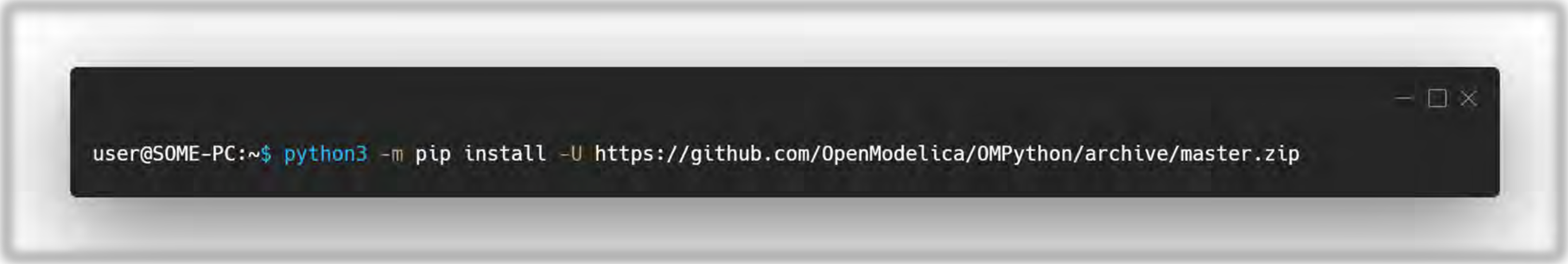
- Install **OMPython**
  - Follow the instructions at github.com/OpenModelica/OMPython
  - For **Windows** + Anaconda Python:
    - Run from Anaconda Prompt (Anaconda3)



```
(base) C:\Users\userName>echo %OPENMODELICAHOME%
C:\Program Files\OpenModelica1.17.0-dev-64bit\

(base) C:\Users\userName>cd %OPENMODELICAHOME%\share\omc\scripts\PythonInterface
(base) C:\Program Files\OpenModelica1.17.0-dev-64bit\share\omc\scripts\PythonInterface>python3 -m pip install -U .
```

Installation Instructions

- Install **OMPython**
  - Follow the instructions at github.com/OpenModelica/OMPython
  - For **Linux**:
    - Use *python3*



```
user@SOME-PC:~$ python3 -m pip install -U https://github.com/OpenModelica/OMPython/archive/master.zip
```

FMI and SPP Standards

Introduction

FMI and SPP Standards

Modelica Association

fmi Functional Mock-Up Interface

ssp System Structure & Parameterization

FMI and SPP Standards

Functional Mock-Up Interface (FMI)

- Free standard
- Defines container and interface to exchange models
- Latest release: FMI 2.0.2
- Latest development build: FMI 3.0 (Alpha)

fmi-standard.org/

fmi Functional Mock-Up Interface

FMI and SPP Standards

Functional Mock-Up Unit (FMU)

- Model Exchange (ME)

> *[…] C code representation of a dynamic system model that can be utilized by other modeling and simulation environments.*

- Co-Simulation (CS)

> *The intention is to provide an interface standard for coupling of simulation tools in a co-simulation environment*

From: *Functional Mock-up Interface for Model Exchange and Co-Simulation, 2020, version 2.0.2*

FMI and SPP Standards

System Structure & Parameterization (SSP)

> *[…] a tool independent standard to define complete systems consisting of one or more FMUs […] including its parameterization that can be transferred between simulation tools.*

From: *https://ssp-standard.org/*

ssp-standard.org/

ssp

System Structure
& Parameterization

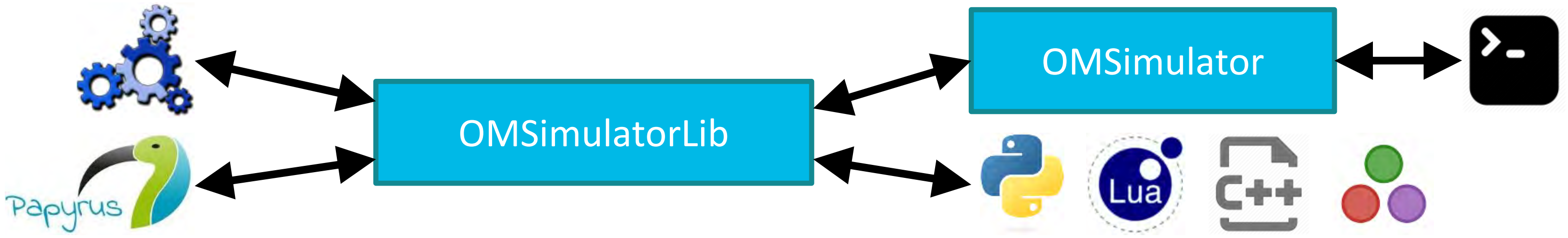
OMSimulator

Introduction

What's new in OMSimulator

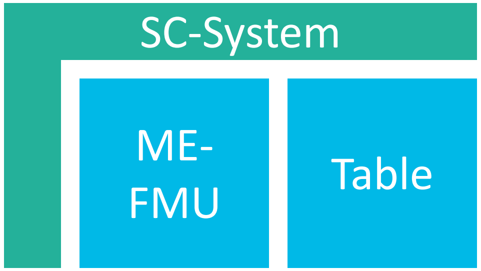
- Released OMSimulator v2.1.1 (Jan 2021)
  - SSP compliant
  - FMI Cross Check
  - Improved graphical user interface (OMEdit)
  - Improved Python interface
  - New non-linear solver Kinsol
  - Bug fixes

User Interface



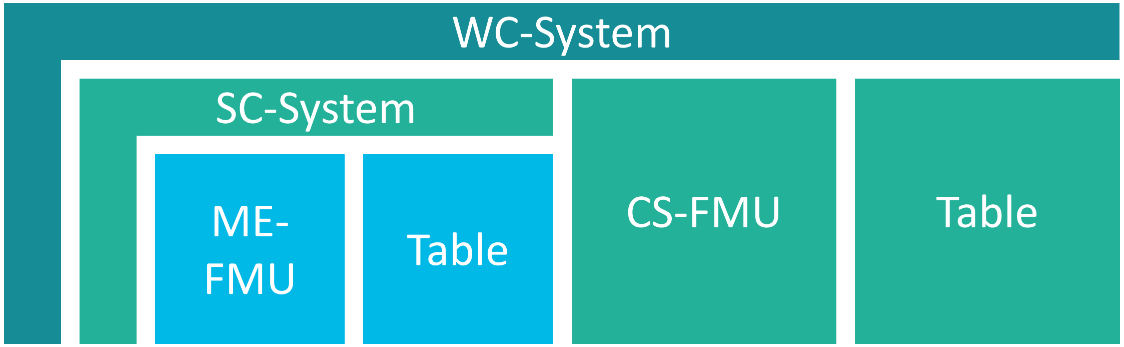
- Command-line interface
- Scripting interface
- Graphical interface

Composite Model Structure (I)



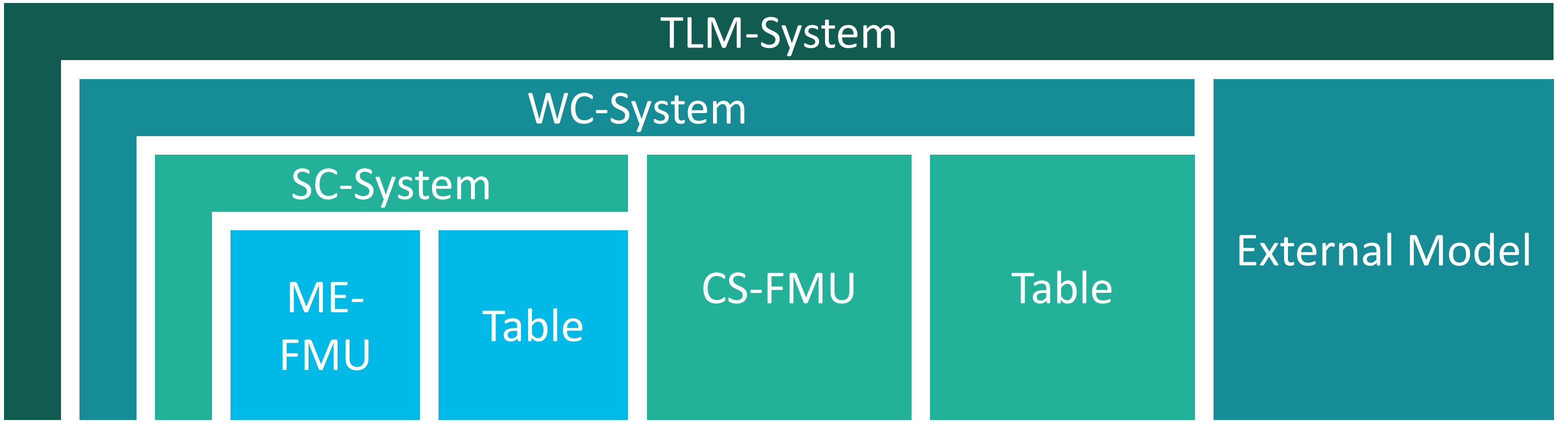
- Strongly Connected System
  - direct communication schema
- Detecting and handling algebraic loops
- Integration methods
  - Explicit euler
  - Cvode

Composite Model Structure (II)



- Weakly connected system
  - Communication at communication time points
  - Extrapolation of inputs

Composite Model Structure (III)



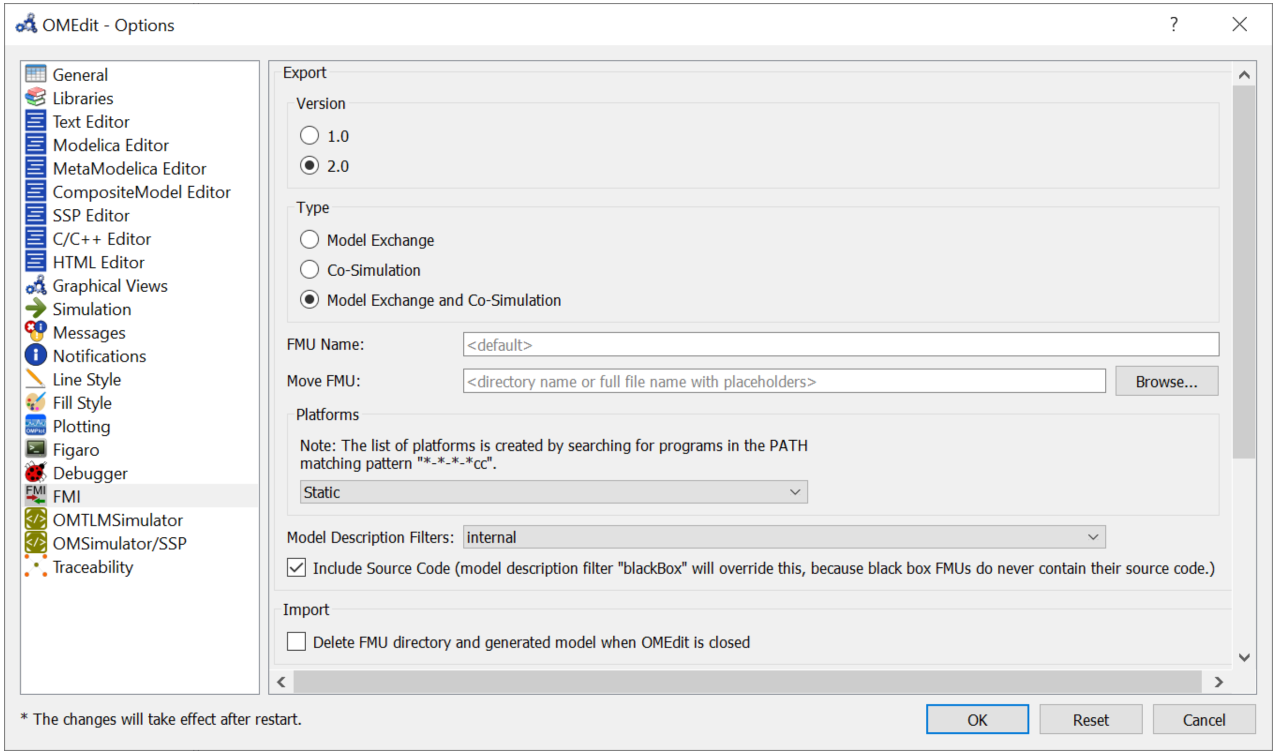
- Transmission Line Modelling
  - Physical signal connections

FMI Export

Introduction

FMI Export

- Check FMI setting in OMEdit (Tools -> Options)



OMEdit - Options

Export

Version

○ 1.0

● 2.0

Type

○ Model Exchange

○ Co-Simulation

● Model Exchange and Co-Simulation

FMU Name: <default>

Move FMU: <directory name or full file name with placeholders>

Platforms

Note: The list of platforms is created by searching for programs in the PATH
matching pattern "*-*-*-*cc".

Static

Model Description Filters: internal

☑ Include Source Code (model description filter "blackBox" will override this, because black box FMUs do never contain their source code.)

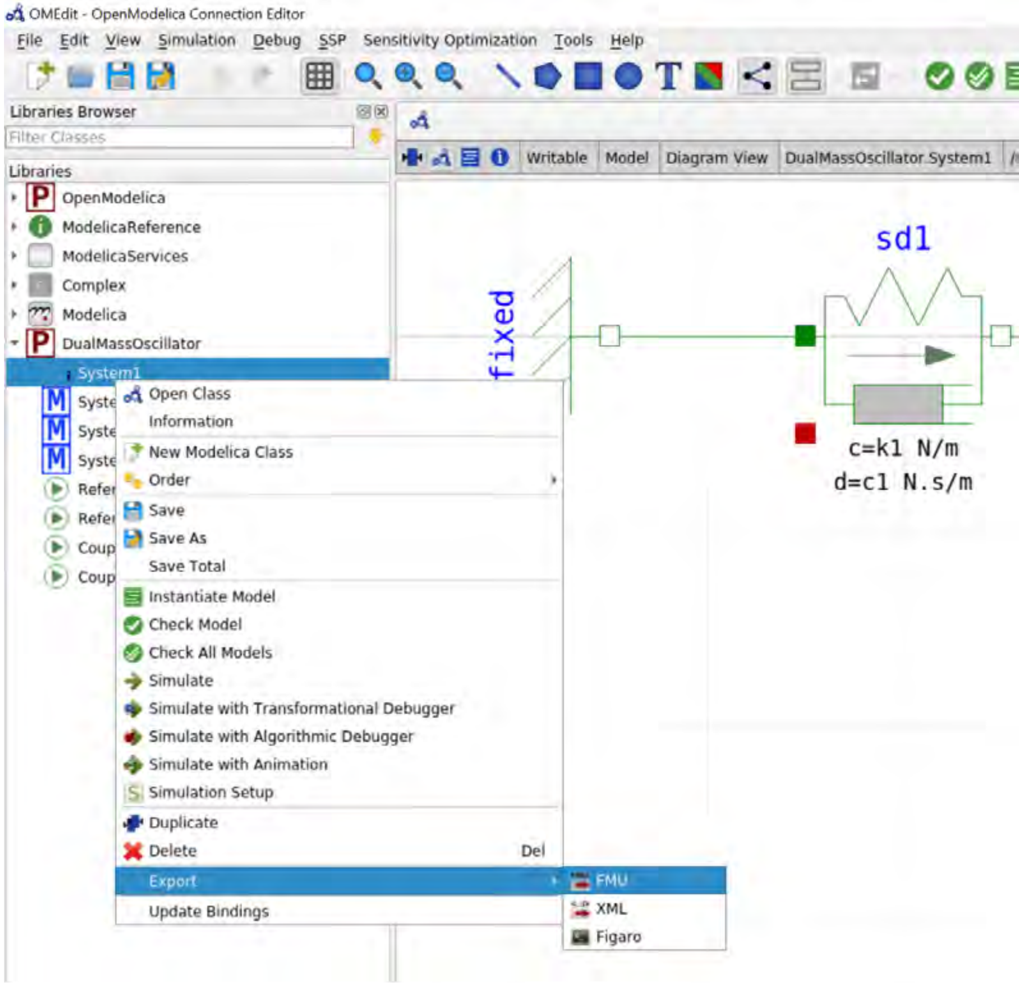
Import

□ Delete FMU directory and generated model when OMEdit is closed

* The changes will take effect after restart.

FMI Export

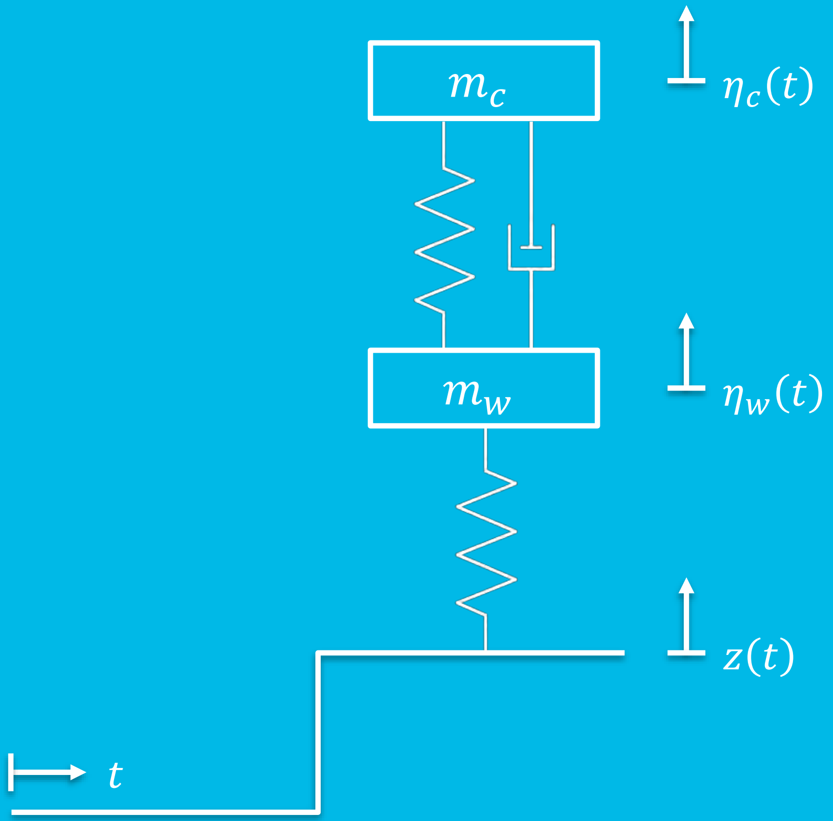
- Open a Modelica model
- Right-click
  Select Export -> FMU



Live demo

Quarter Car Model

Exercise

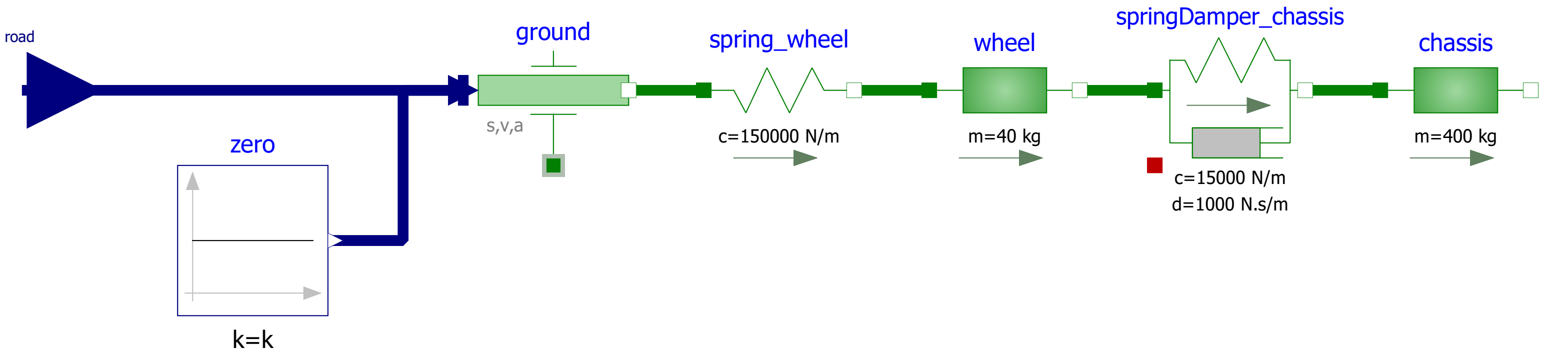


Quarter Car Model - Jupyter Notebook

- Simulating a single FMU with OMSimulator
- CSV input to FMU
- Python scripting with OMSimulator Python interface

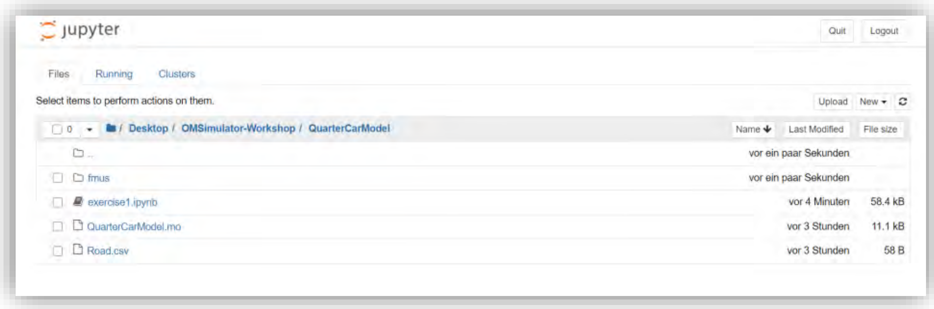
Quarter Car Model - Jupyter Notebook

- Use Jupyter Notebook to open **QuarterCarModel /exercise1.ipynb** and start hacking!
- Install instructions can be found at the beginning of the presentation



Quarter Car Model - Jupyter Notebook

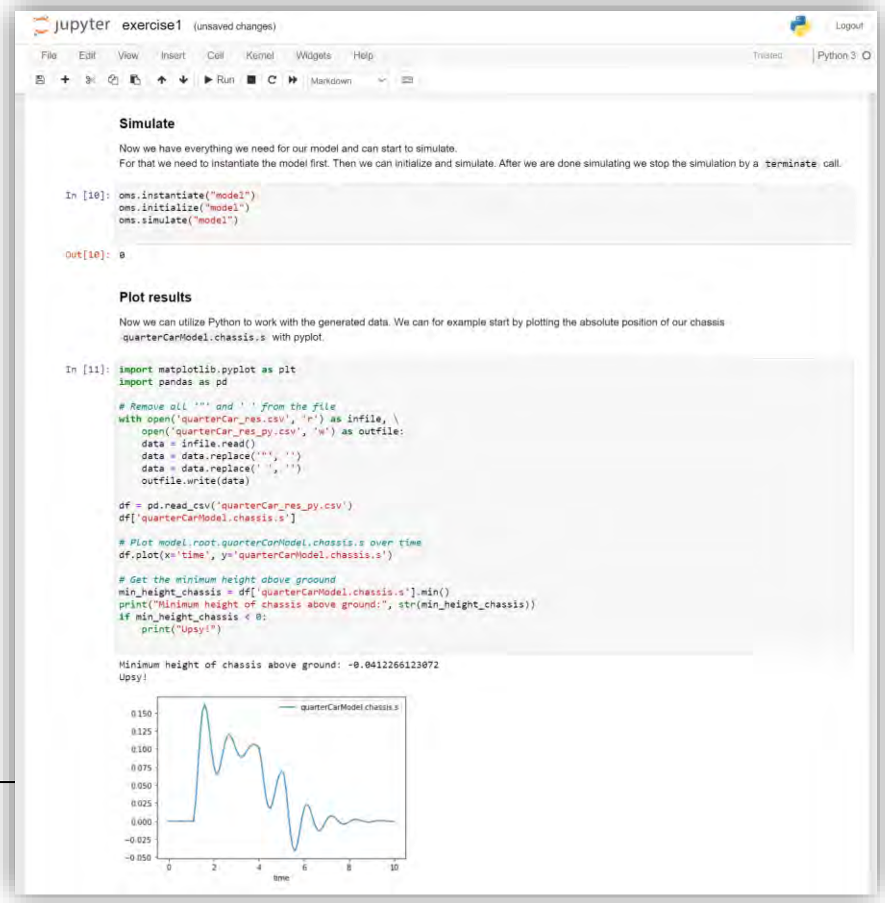
- In Jupyter navigate to *exercise1.ipynb*
- Have fun!



Select items to perform actions on them.

□ 0 ▾ / Desktop / OMSimulator-Workshop / QuarterCarModel

Name ↓	Last Modified	File size
..	vor ein paar Sekunden	
□ fmus	vor ein paar Sekunden	
□ exercise1.ipynb	vor 4 Minuten	58.4 kB
□ QuarterCarModel.mo	vor 3 Stunden	11.1 kB
□ Road.csv	vor 3 Stunden	58 B



Simulate

Now we have everything we need for our model and can start to simulate.
For that we need to instantiate the model first. Then we can initialize and simulate. After we are done simulating we stop the simulation by a `terminate` call.

```
In [10]: oms.instantiate("model")
         oms.initialize("model")
         oms.simulate("model")

Out[10]: 0
```

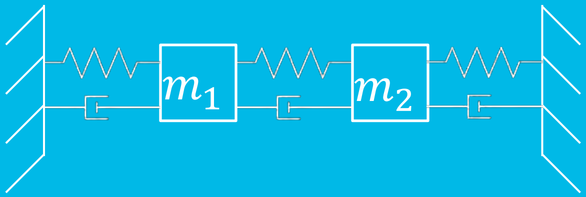
Plot results

Now we can utilize Python to work with the generated data. We can for example start by plotting the absolute position of our chassis `quarterCarModel.chassis.s` with pyplot.

```
In [11]: import matplotlib.pyplot as plt
         import pandas as pd

         # Remove all '"' and ' ' from the file
         with open('quarterCar_res.csv', 'r') as infile, \
             open('quarterCar_res_py.csv', 'w') as outfile:
             data = infile.read()
             data = data.replace('"', '')
             data = data.replace(' ', '')
             outfile.write(data)

         df = pd.read_csv('quarterCar_res_py.csv')
         df['quarterCarModel.chassis.s']

         # Plot model.root.quarterCarModel.chassis.s over time
         df.plot(x='time', y='quarterCarModel.chassis.s')

         # Get the minimum height above groound
         min_height_chassis = df['quarterCarModel.chassis.s'].min()
         print("Minimum height of chassis above ground:", str(min_height_chassis))
         if min_height_chassis < 0:
             print("Upsy!")

Minimum height of chassis above ground: -0.0412266123072
Upsy!
```

Dual Mass Oscillator

Exercise

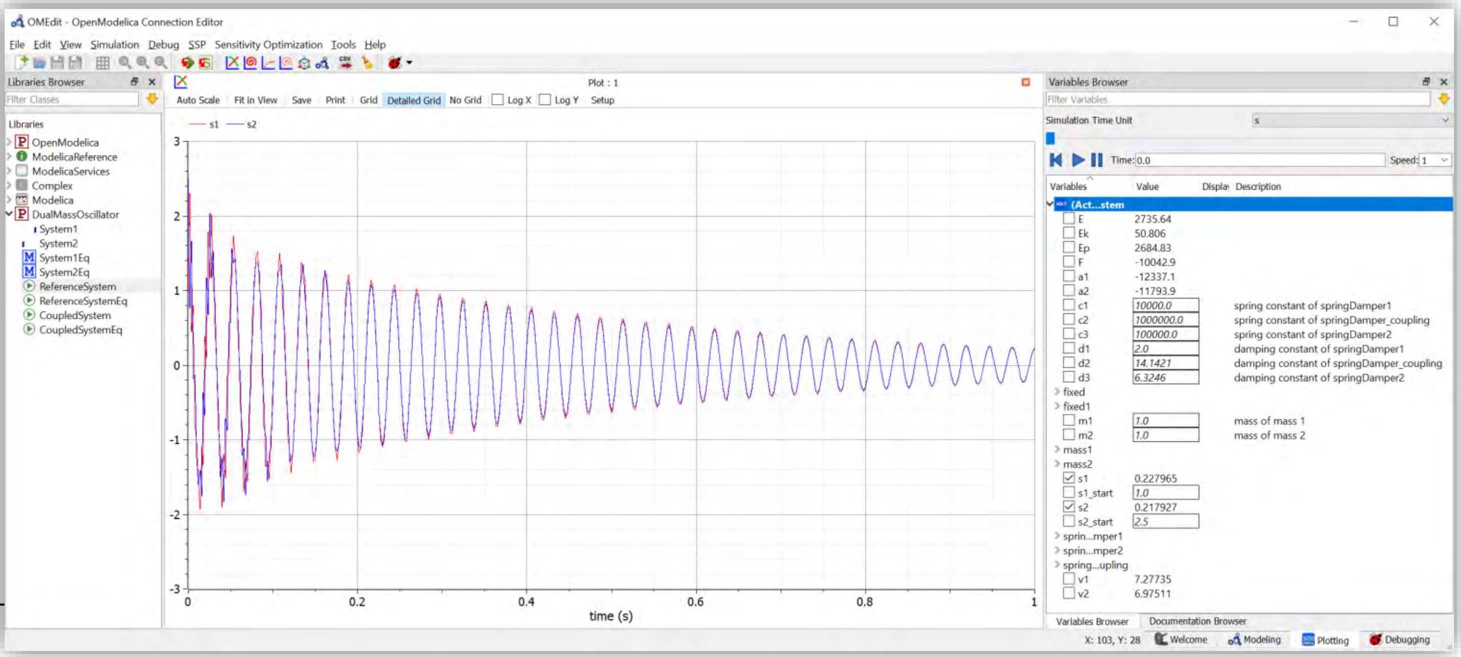


Dual Mass Oscillator

- Splitting the mechanical (reference) model into two subsystems using force-displacement coupling
- Defining interfaces for the FMUs
- Creating a FMU-based composite model (CS/ME)
- Set start values
- Simulate the composite model
- Export as SSP model

Dual Mass Oscillator (I)

- Open `DualMassOscillator.mo` in OMEdit
- Simulate `DualMassOscillator.ReferenceSystem`
- Perturb the system with `s1_start` and `s2_start`



Simulation Time Unit: s

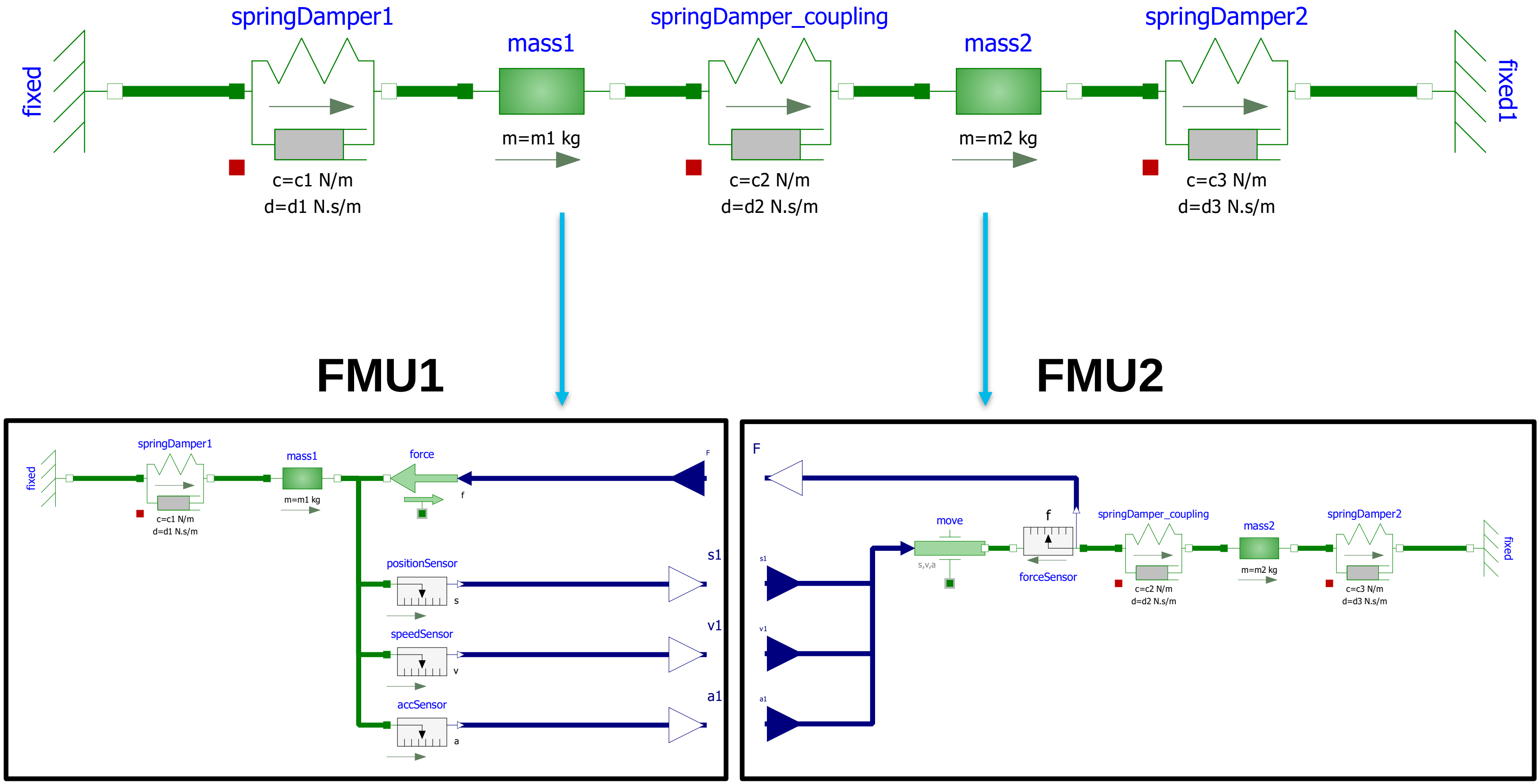
Time: 0.0 Speed: 1

Variables	Value	Display	Description
(Act...stem			
□ E	2735.64		
□ Ek	50.806		
□ Ep	2684.83		
□ F	-10042.9		
□ a1	-12337.1		
□ a2	-11793.9		
□ c1	10000.0		spring constant of springDamper1
□ c2	1000000.0		spring constant of springDamper_coupling
□ c3	100000.0		spring constant of springDamper2
□ d1	2.0		damping constant of springDamper1
□ d2	14.1421		damping constant of springDamper_coupling
□ d3	6.3246		damping constant of springDamper2
fixed			
fixed1			
□ m1	1.0		mass of mass 1
□ m2	1.0		mass of mass 2
mass1			
mass2			
☑ s1	0.227965		
□ s1_start	1.0		
☑ s2	0.217927		
□ s2_start	2.5		
sprin...mper1			
sprin...mper2			
spring...upling			
□ v1	7.27735		
□ v2	6.97511		

Dual Mass Oscillator (II)

- Break the model `DualMassOscillator.ReferenceSystem` down into two FMUs
  - Note: Duplicate this model and delete the not needed components
- Define interfaces (inputs/outputs) by adding signal ports from `Blocks.Interfaces` and sensors e.g. from `Electrical.Analog.Sensors`

Dual Mass Oscillator (II)



Dual Mass Oscillator (III)

- Use Jupyter Notebook to open **DualMassOscillator /exercise2.ipynb**
- Do part III of the exercise to:
  - Export FMUs with OMPython
  - Create ME CS FMUs
  - (optional) Export CS FMUs with CVODE integrator

Dual Mass Oscillator (IV)

- Use Jupyter Notebook to open **DualMassOscillator /exercise2.ipynb**
- Do part IV of the exercise to:
  - Import FMUs
  - Create strongly coupled systems
  - Set start values and simulate models
  - See differences between strongly and weekly coupled systems

Wrap-up/Questions