

Tutorial 2

FMI for Composite Modelling, Co-Simulation and Model Exchange

Andreas Heuermann and Lennart Ochel

15th MODPROD Workshop, February 3-4, 2021

Outline

- Preparation
 - Installation instructions
- Introduction
 - FMI and SSP standards
 - OMSimulator
- Exercises
 - Quarter Car Model
 - Dual Mass Oscillator

Installation Instructions

Preparation

Installation Instructions

What you will need for this tutorial:

- **OpenModelica** $\geq v1.17.0$ -dev installed
- **Python3** (version ≥ 3.8) installed with modules
 - OMPython
 - OMSimulator (version $\geq 2.1.1$)
- **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 \geq v1.17.0-dev
 - OMSimulator is part of **OMEdit**
 - GUI + CLI + scripting available
 - Follow instructions for your platform

openmodelica.org

OpenModelica

Installation Instructions



The image shows the OpenModelica website header and a section for downloading Windows versions. The header features the OpenModelica logo on the left and 'Login' and 'Create an account' links on the right. Below the header is a dark navigation bar with links: HOME, DOWNLOAD, TOOLS & APPS, USERS, DEVELOPERS, FORUM, EVENTS, and RESEARCH. A search bar is located on the right side of the navigation bar. The main content area is titled 'Download Windows' and lists three download options: Official Release, Stable Development, and Nightly Build. Each option includes a version number, bitness, and a list of features or release details.

| Download Type | Version | Details |
|---------------------------|-----------------------------|--|
| Official Release | 1.16.2 (32bit/64bit) | <ul style="list-style-type: none">contains only validated new featuresintended for productive usage (commit history) |
| Stable Development | 1.16.2 (32bit/64bit) | <ul style="list-style-type: none">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 validateddev.betaxx versions are released in preparation to official releases for testing; no new features are added to beta versions, only bug fixeslatest stable release: 1.16.2 (commit history) |
| Nightly Build | 1.17.0-dev (32bit/64bit) | <ul style="list-style-type: none">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 usagefeatures that are not subject to regression testing may get broken between one nightly build and the next |

Installation Instructions

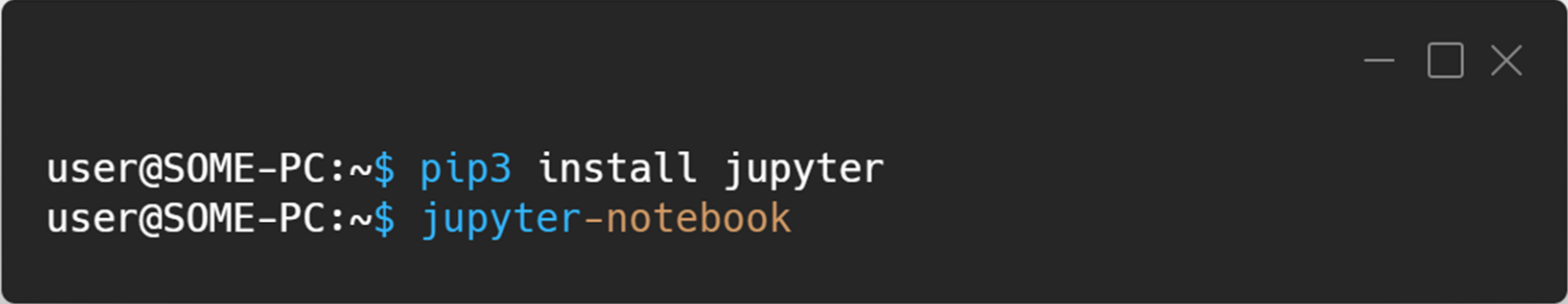
Install **Jupyter Notebook** on Windows:

- 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

A dark-themed terminal window with standard window controls (minimize, maximize, close) in the top right corner. It contains two lines of text: a command to install Jupyter and the command to launch the Jupyter Notebook interface.

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

Installation Instructions

Install **OMSimulator** (version $\geq 2.1.1$) with pip

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


pip3 install OMSimulator

A terminal window with a dark background and light gray window controls (minimize, maximize, close) in the top right corner. The text inside the terminal is 'user@SOME-PC:~\$ pip3 install OMSimulator', where 'pip3' is highlighted in blue.

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

Installation Instructions

- Install **OMP**ython
 - Follow the instructions at github.com/OpenModelica/OMPpython
 - 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



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 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/



System Structure
& Parameterization

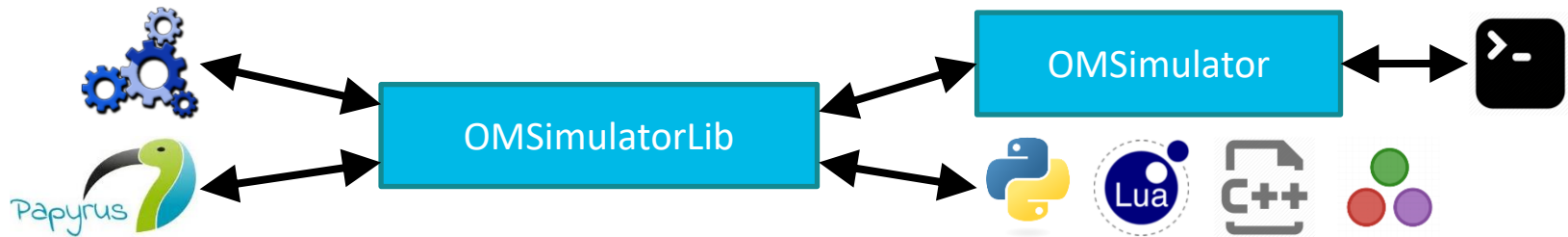
OMSimulator

Introduction

What's new in OMSimulator

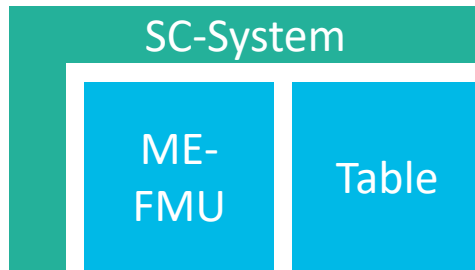
- Released OMSimulator v2.1.1 (Jan 2021)
 - SSP compliant
 - FMI Cross Check
 - Improved graphical user interface (OMEdit)
 - Improved Python interface
 - New non-linear solver Kinsol
 - Bug fixes

User Interface



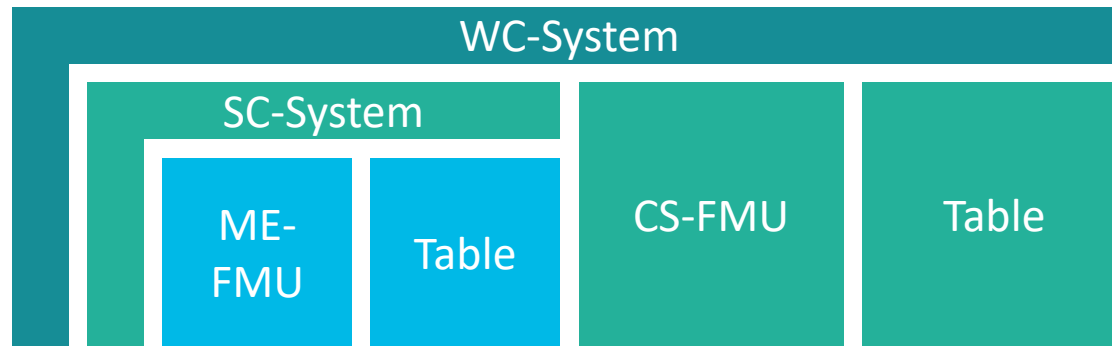
- Command-line interface
- Scripting interface
- Graphical interface

Composite Model Structure (I)



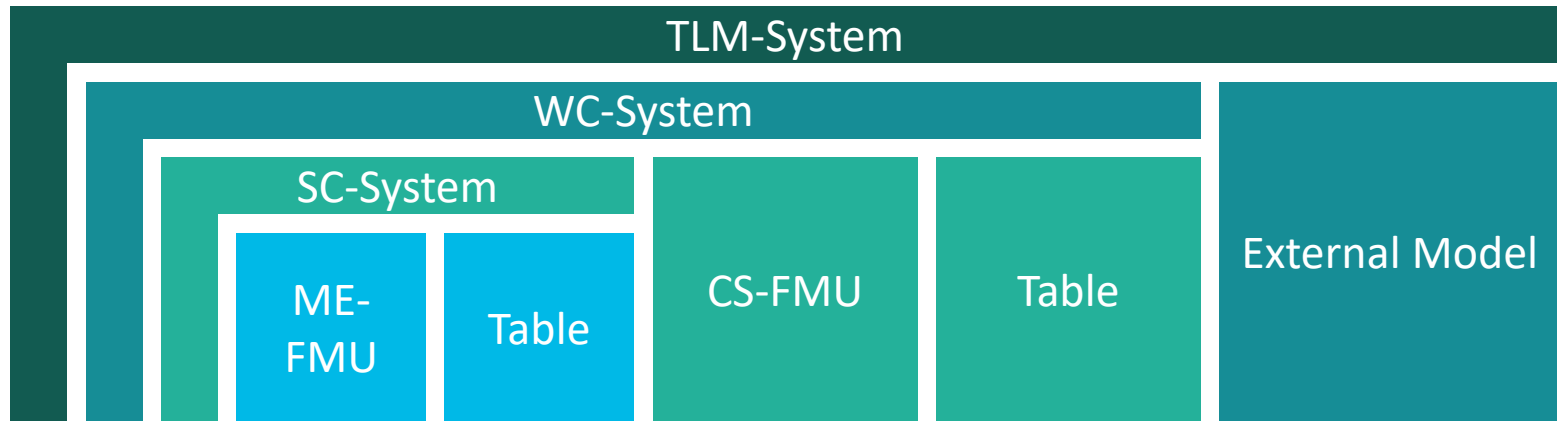
- Strongly Connected System
 - direct communication schema
- Detecting and handling algebraic loops
- Integration methods
 - Explicit euler
 - Ccode

Composite Model Structure (II)



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

Composite Model Structure (III)



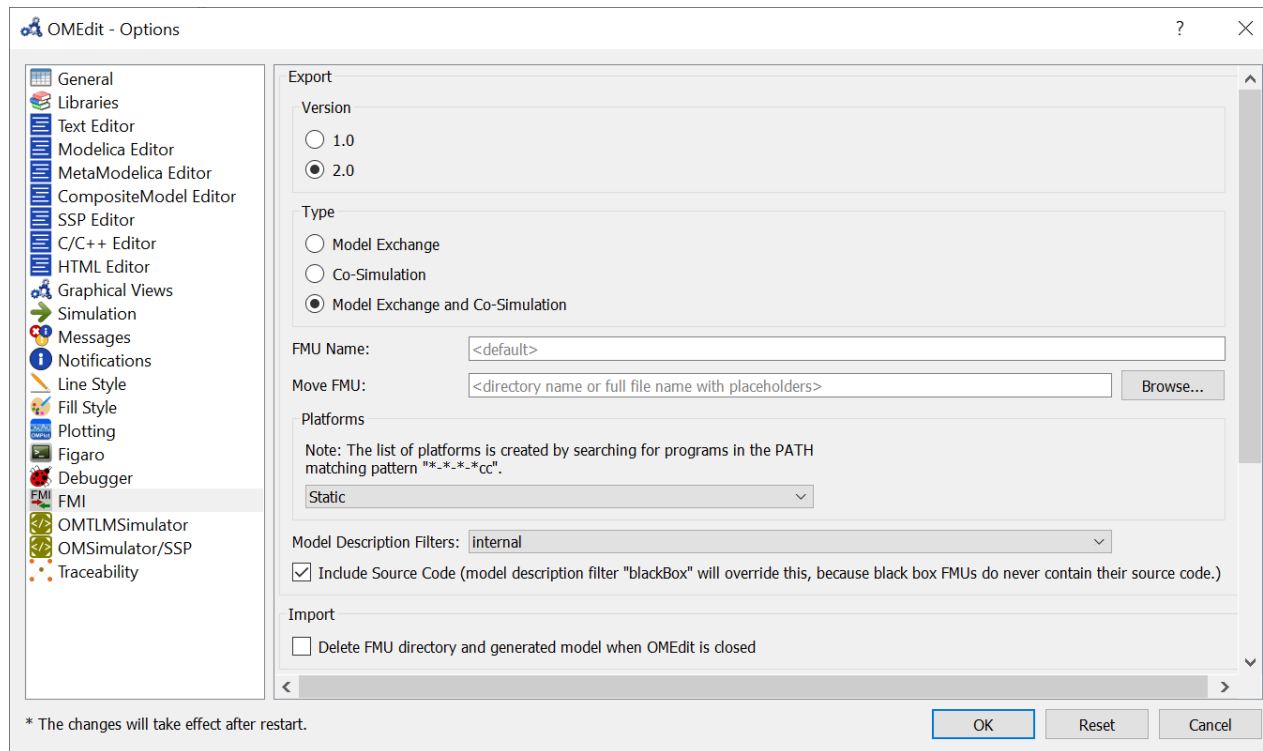
- Transmission Line Modelling
 - Physical signal connections

FMI Export

Introduction

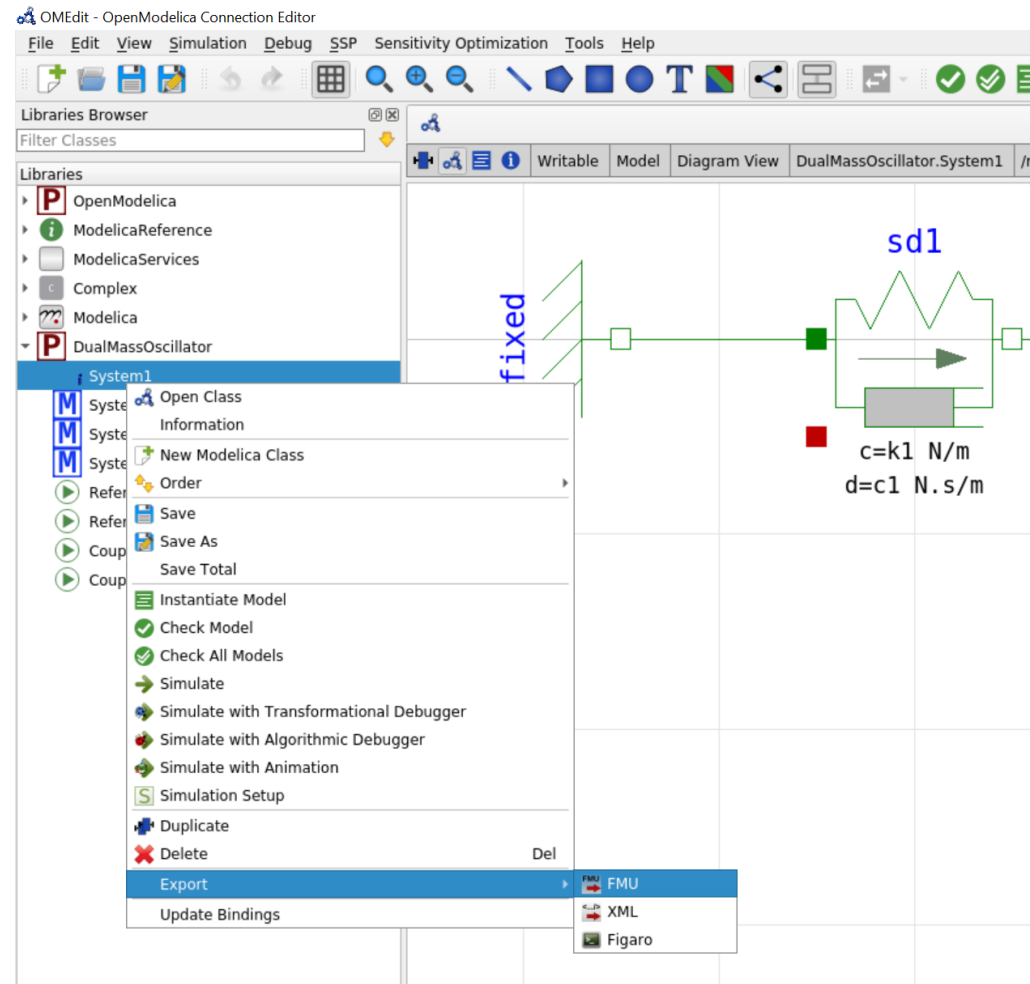
FMI Export

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



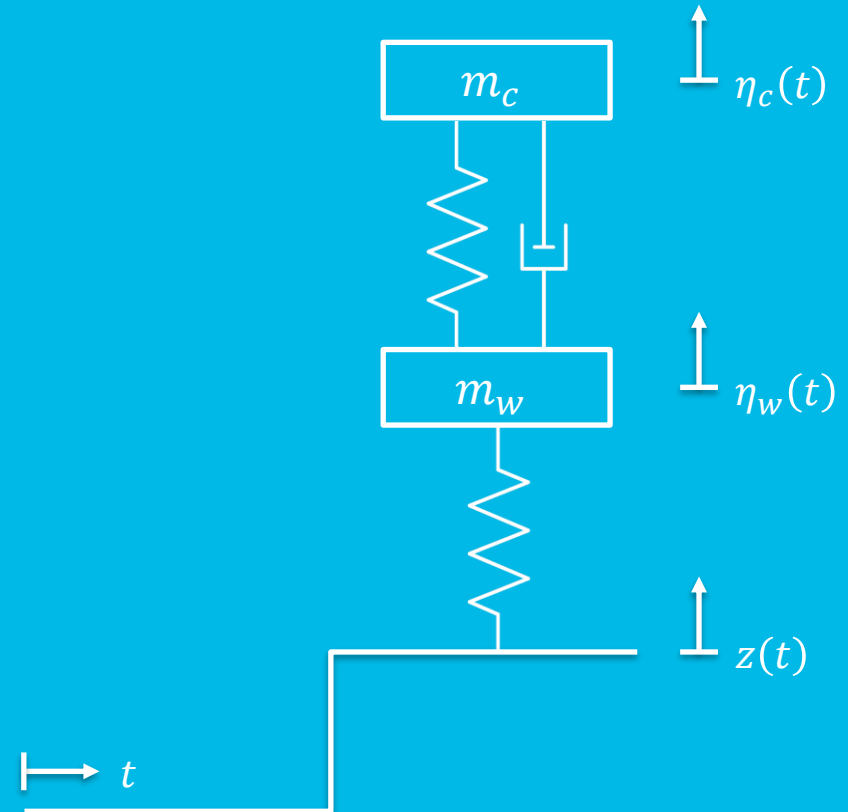
FMI Export

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



Quarter Car Model

Exercise

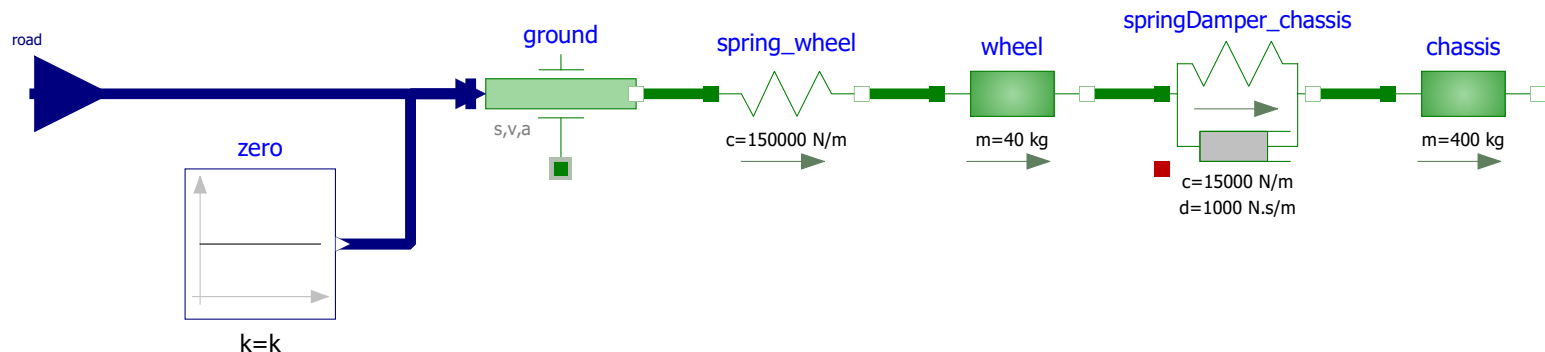


Quarter Car Model - Jupyter Notebook

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

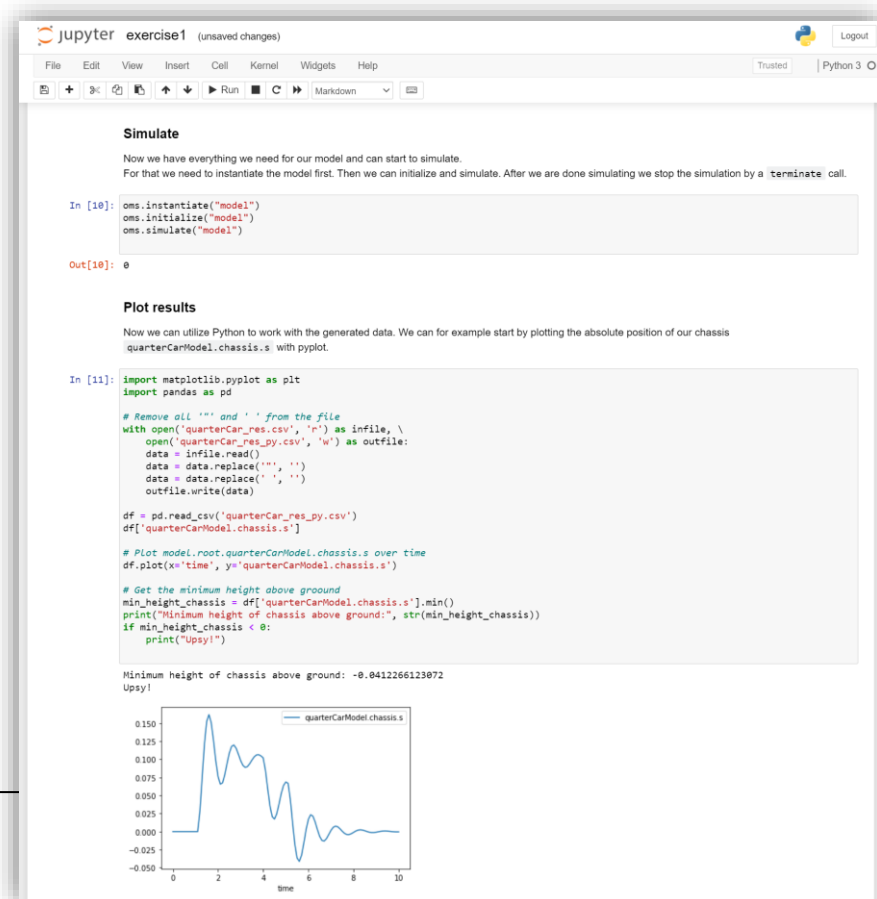
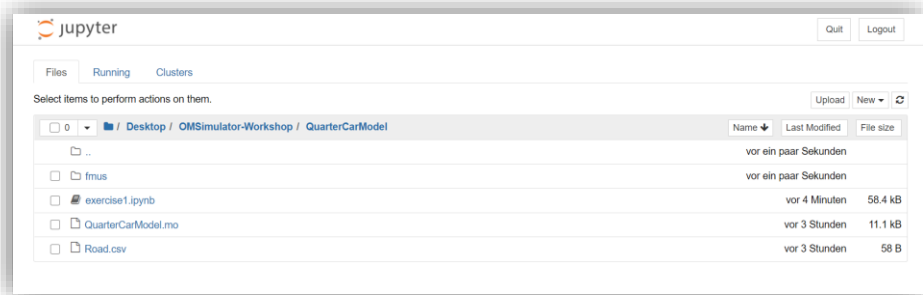
Quarter Car Model - Jupyter Notebook

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



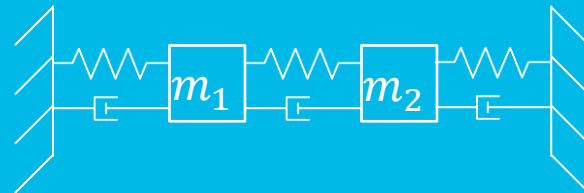
Quarter Car Model - Jupyter Notebook

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



Dual Mass Oscillator

Exercise

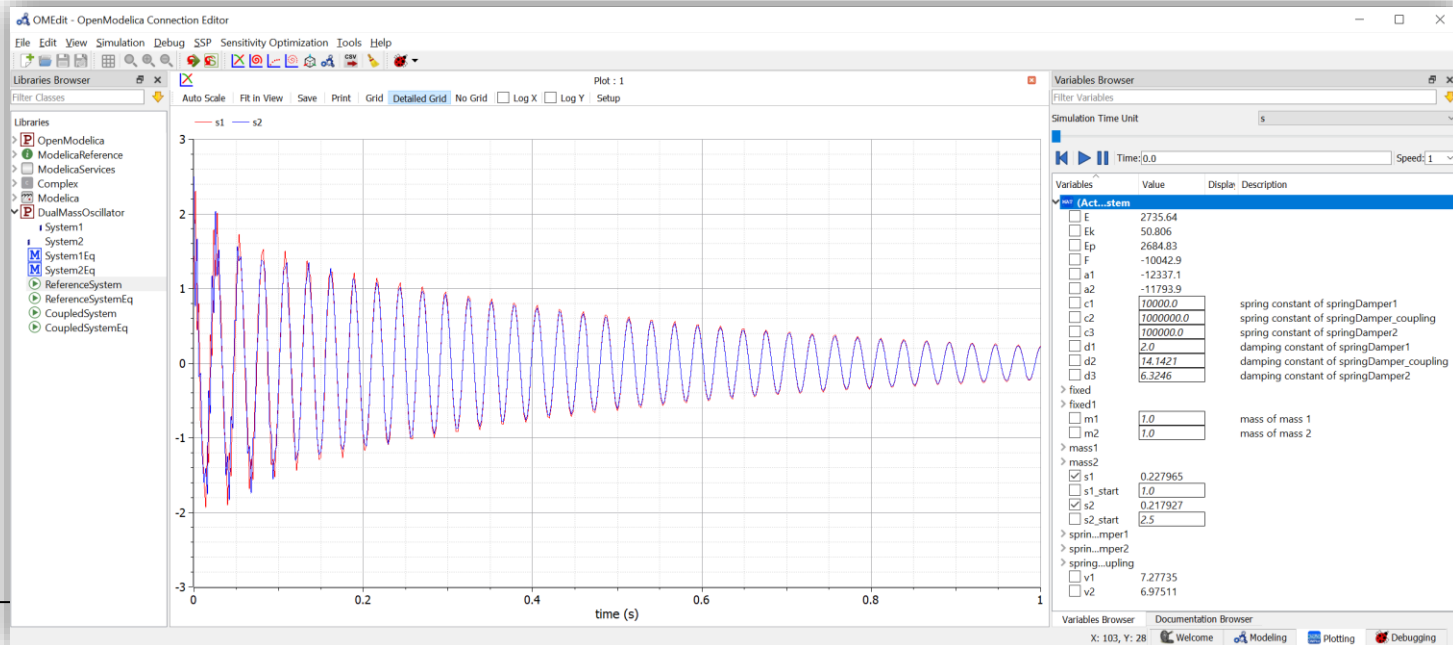


Dual Mass Oscillator

- 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)

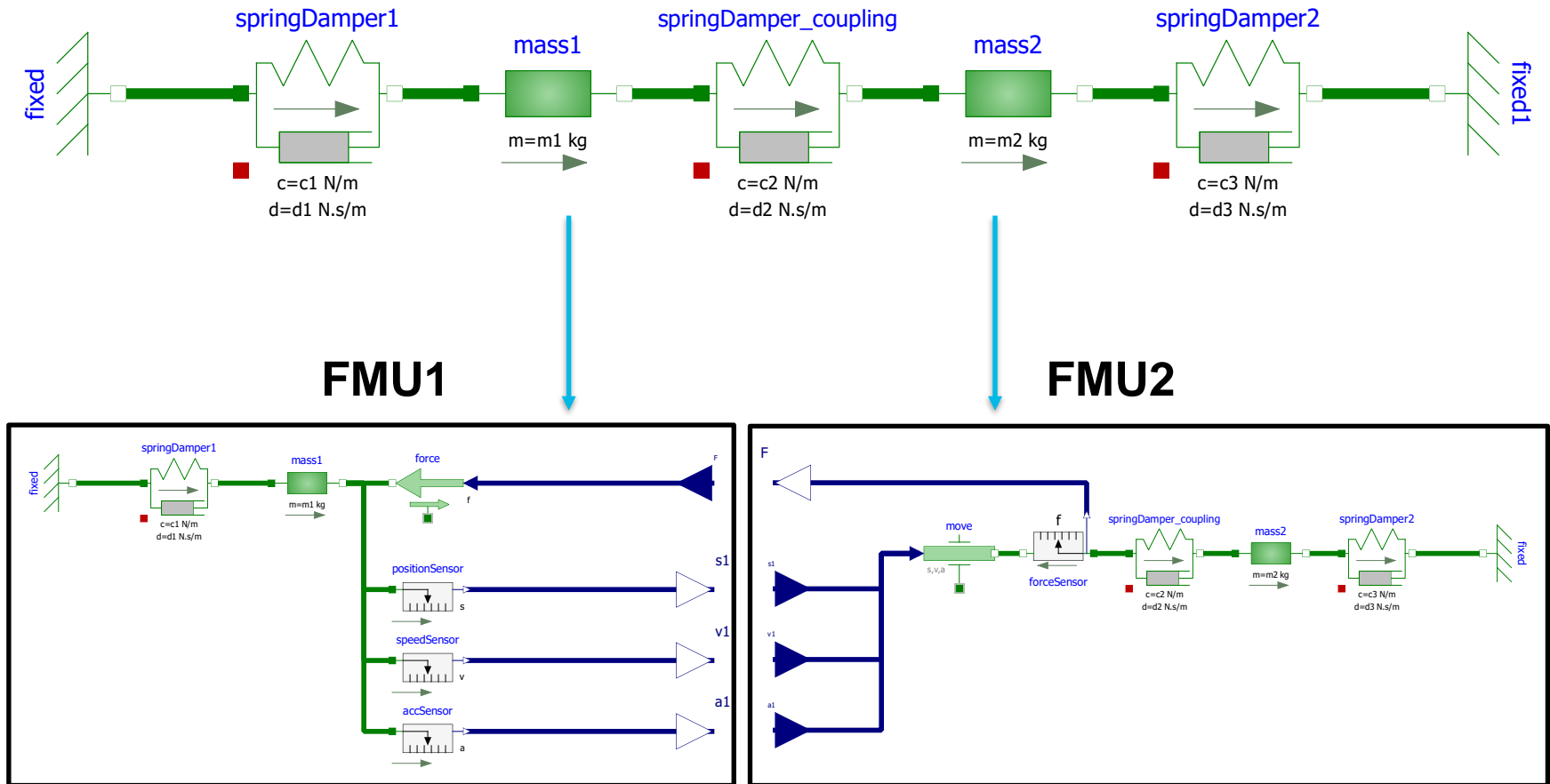
- Open DualMassOscillator.mo in OMEdit
- Simulate DualMassOscillator.ReferenceSystem
- Perturb the system with s1_start and s2_start



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 weakly coupled systems