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
- Exercises
  - Quarter Car Model
  - Dual Mass Oscillator

Installation Instructions

Preparation

Installation Instructions

What you will need for this tutorial:

- **OpenModelica** >=v1.17.0-dev installed
- **Python3** (version >= 3.8) 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

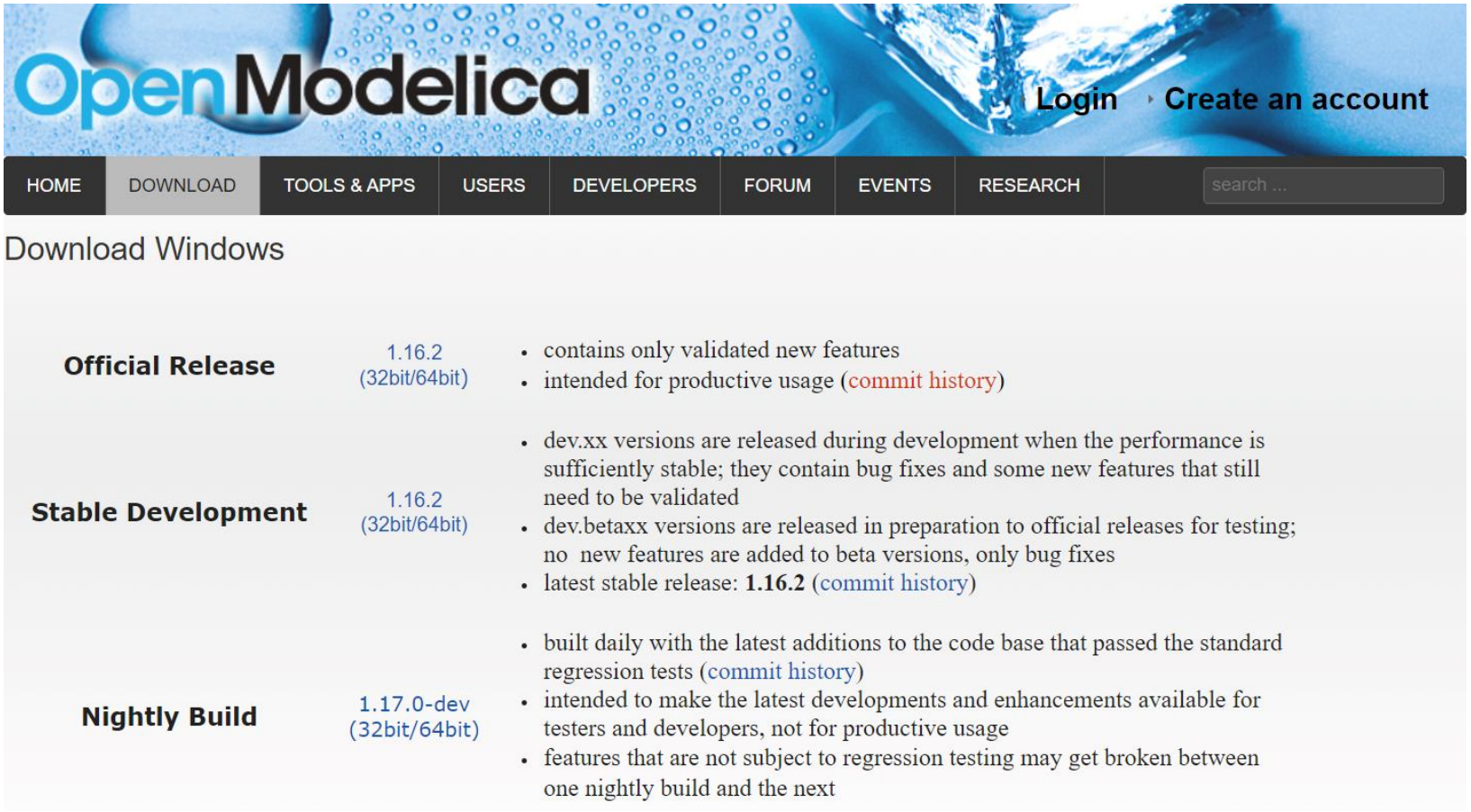
- Documentation
  - OpenModelica User's Guide
    openmodelica.org/doc/OpenModelicaUsersGuide/latest/
  - OMSimulator User's Guide
    openmodelica.org/doc/OMSimulator/master/html/
- Tickets (feature request & bug report)
  - Trac trac.openmodelica.org/OpenModelica/
  - GitHub github.com/OpenModelica/OMSimulator/
- 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

Installation Instructions



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

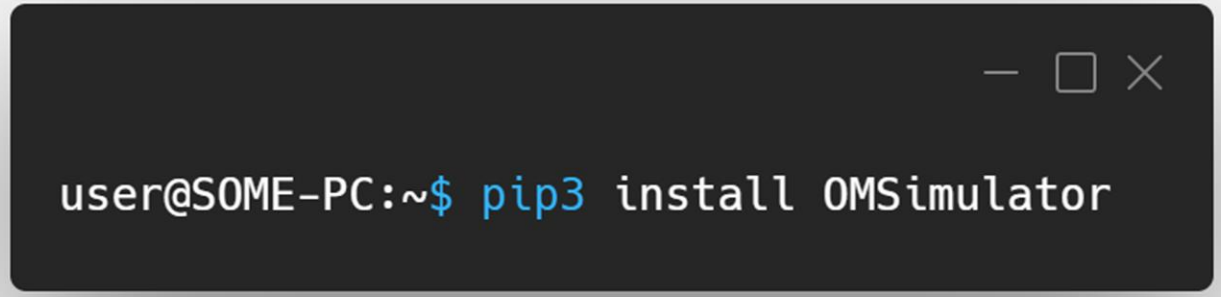
```
user@SOME-PC:~$ pip3 install jupyter
user@SOME-PC:~$ jupyter-notebook
```

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



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

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 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

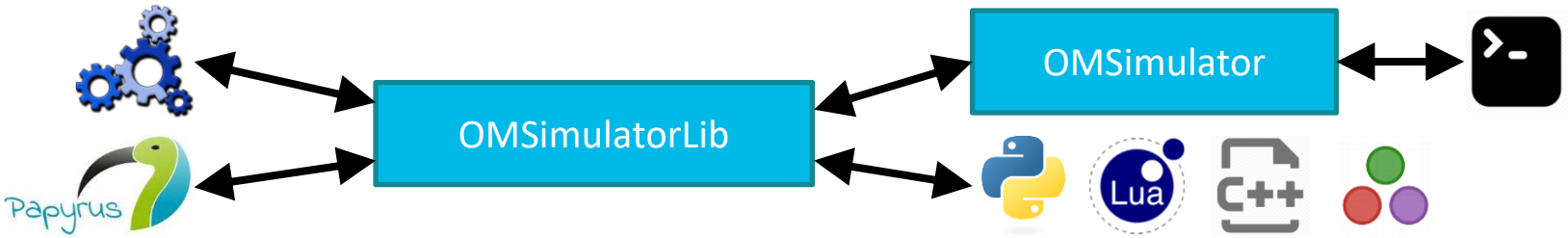
OMSimulator

Introduction

What's new in OMSimulator

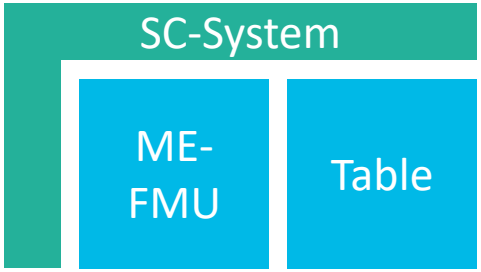
- Released OMSimulator v2.1.1 (Jan 2021)
  - SSP compliant
  - FMI Cross Check
  - Improved graphical user interface (OMEdit)
  - Improved Python interface
  - New non-linear solver Kinsol
  - Bug fixes

User Interface



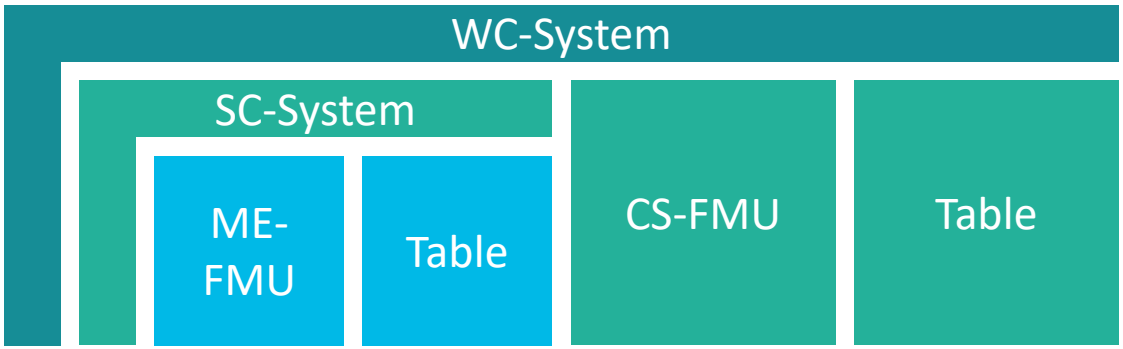
- Command-line interface
- Scripting interface
- Graphical interface

Composite Model Structure (I)



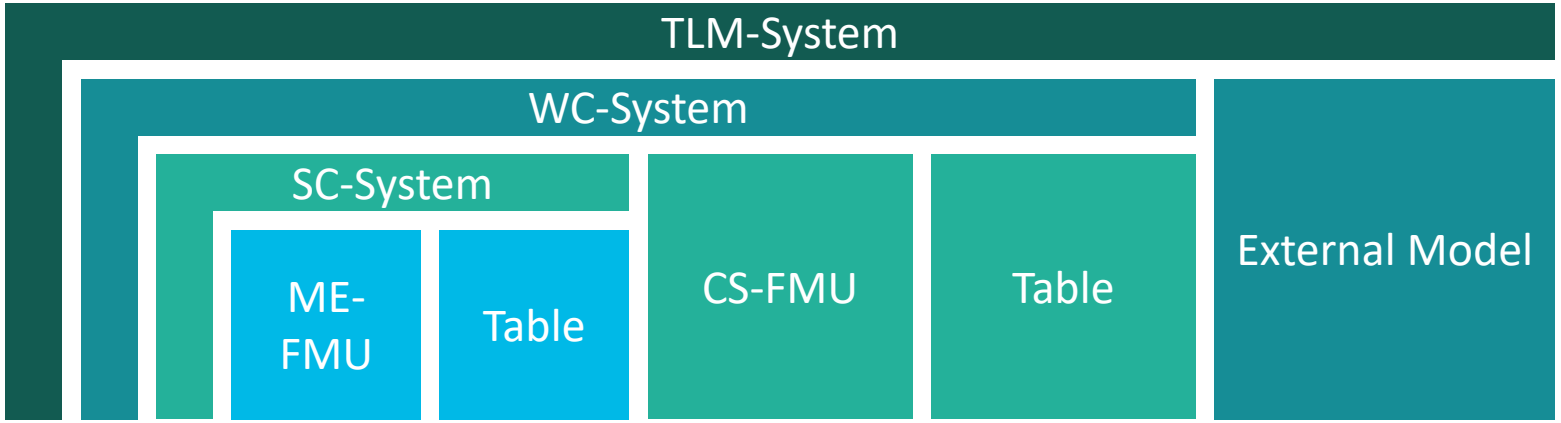
- Strongly Connected System
  - direct communication schema
- Detecting and handling algebraic loops
- Integration methods
  - Explicit euler
  - Cvode

Composite Model Structure (II)



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

Composite Model Structure (III)



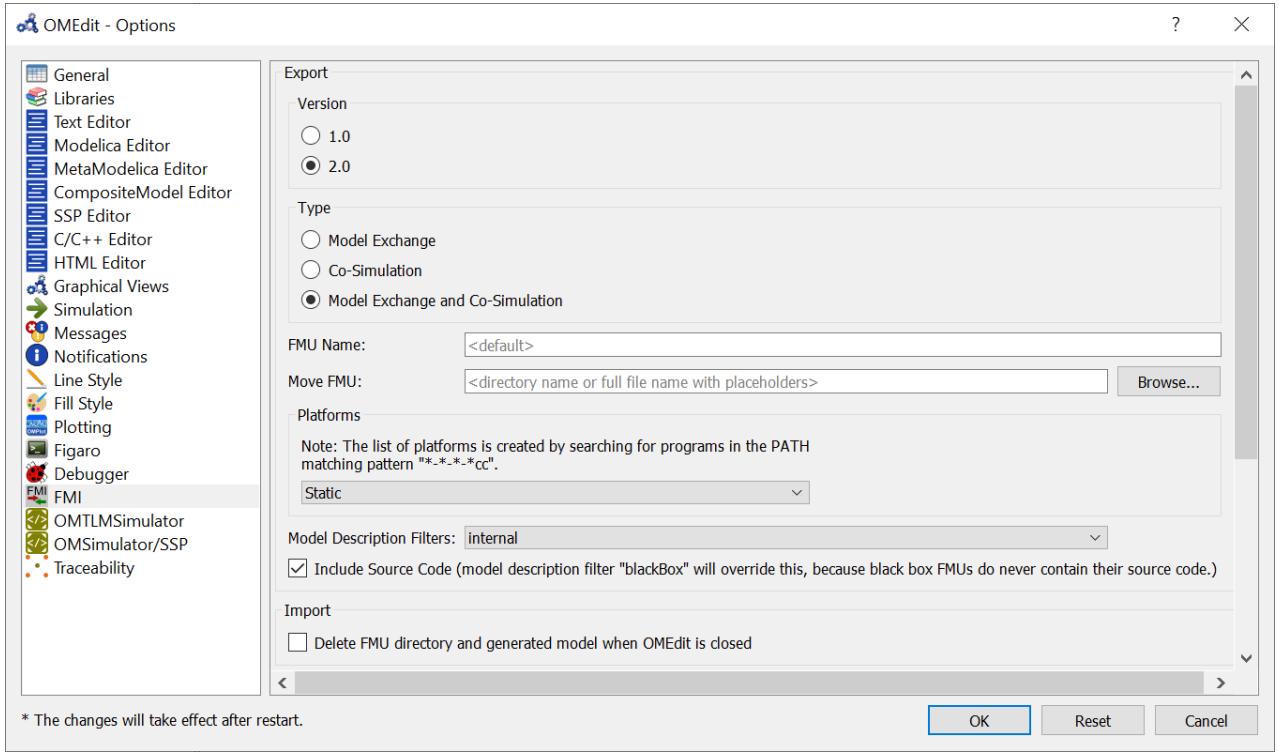
- Transmission Line Modelling
  - Physical signal connections

FMI Export

Introduction

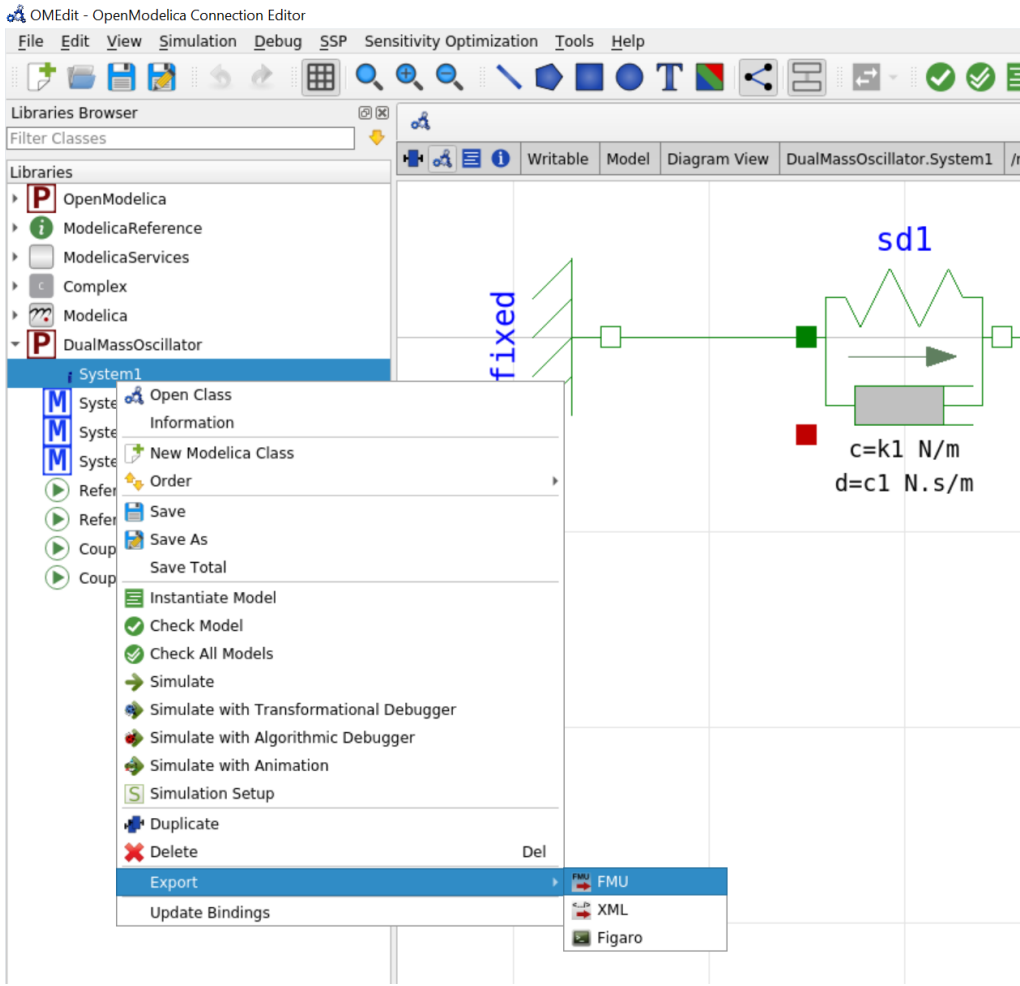
FMI Export

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



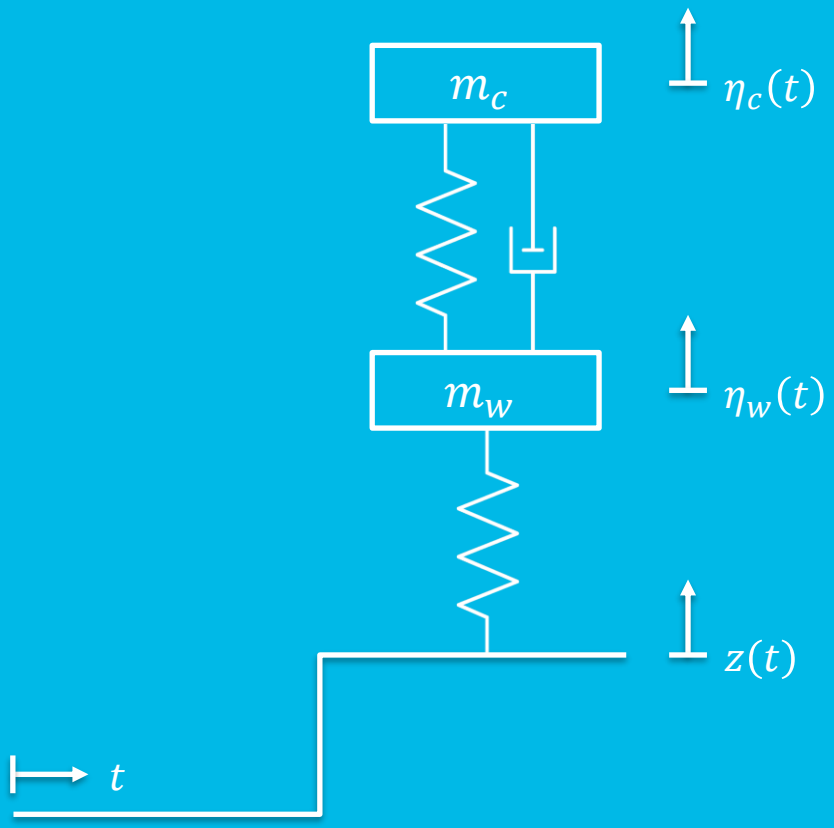
FMI Export

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



Quarter Car Model

Exercise

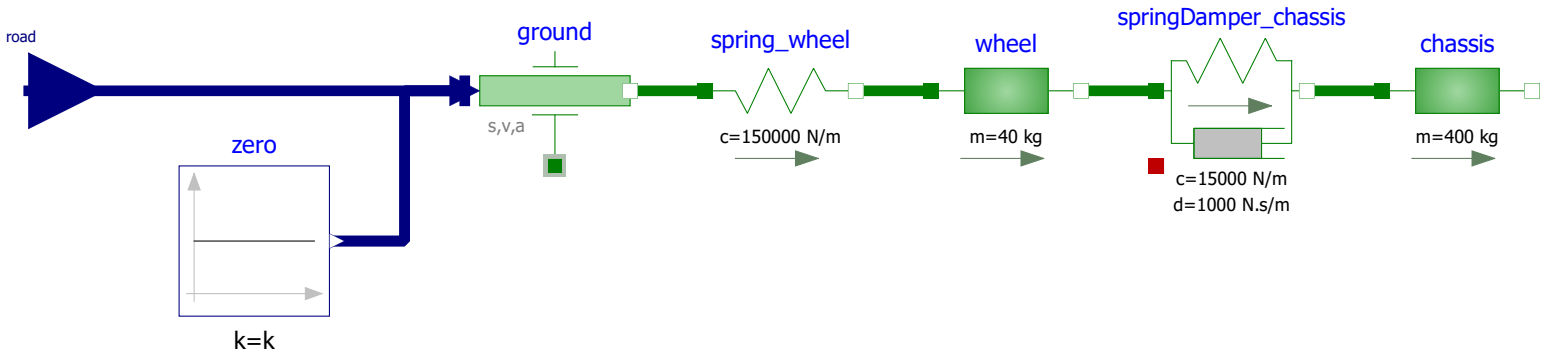


Quarter Car Model - Jupyter Notebook

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

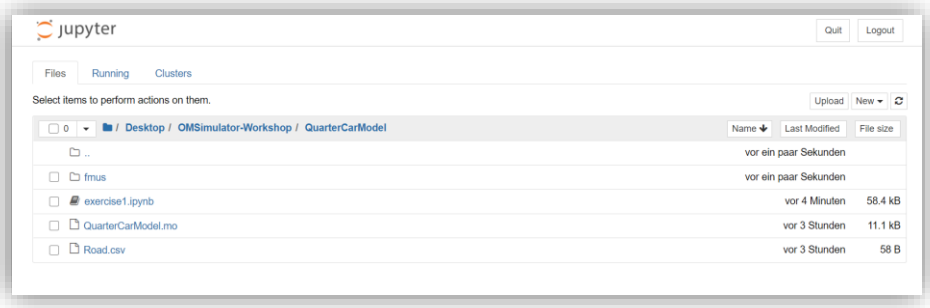
Quarter Car Model - Jupyter Notebook

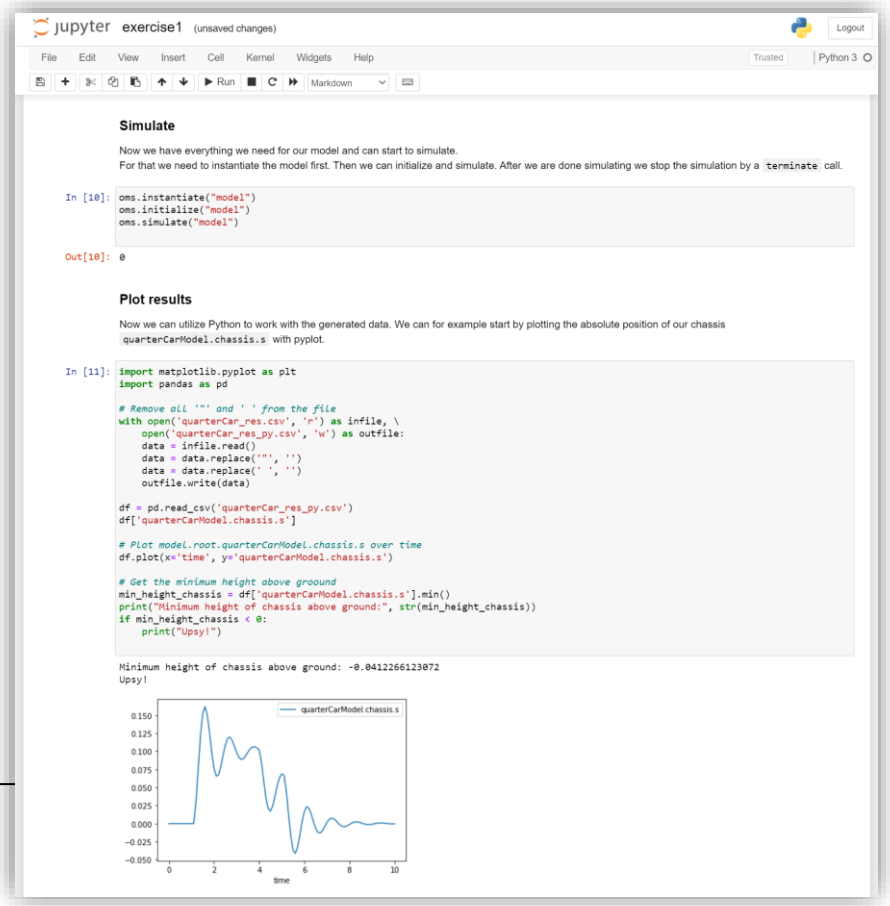
- Use Jupyter Notebook to open **QuarterCarModel /exercise1.ipynb** and start hacking!
- Install instructions can be found at the beginning of the presentation



Quarter Car Model - Jupyter Notebook

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

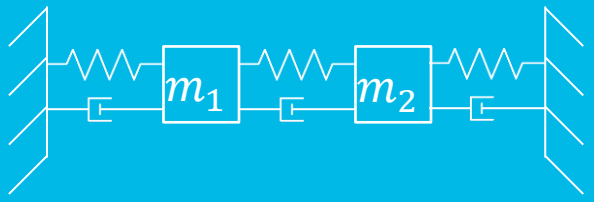




LINKÖPING UNIVERSITY

Dual Mass Oscillator

Exercise

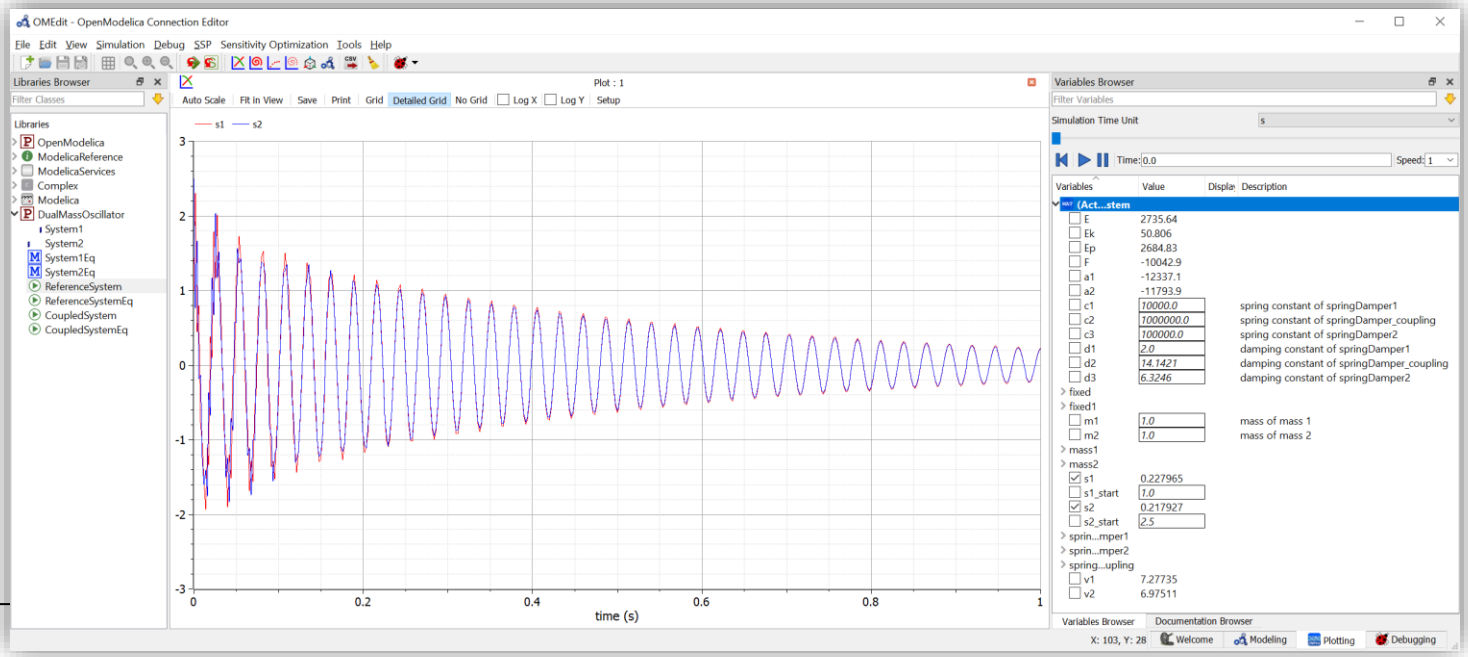


Dual Mass Oscillator

- 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)

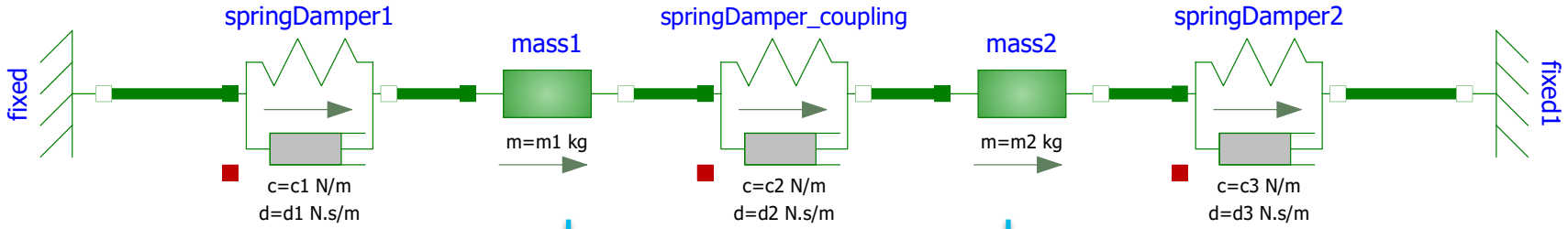
- Open `DualMassOscillator.mo` in OMEdit
- Simulate `DualMassOscillator.ReferenceSystem`
- Perturb the system with `s1_start` and `s2_start`



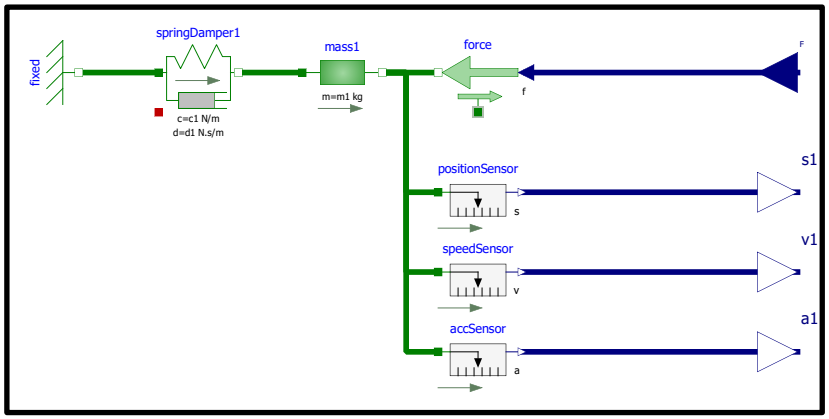
Dual Mass Oscillator (II)

- 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`

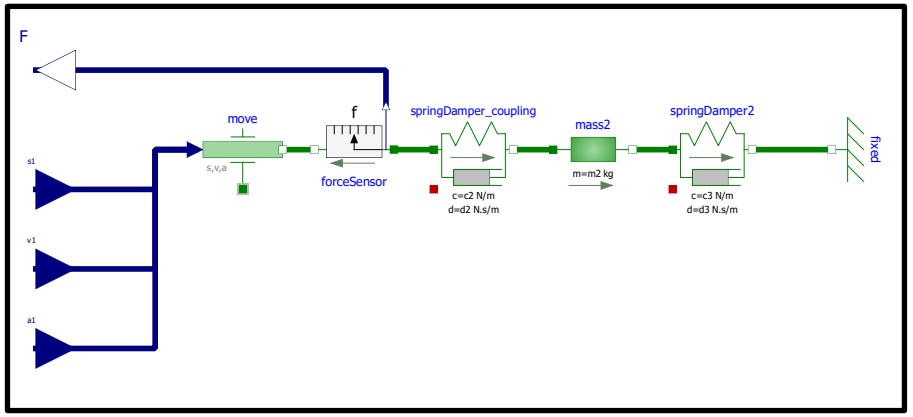
Dual Mass Oscillator (II)



FMU1



FMU2



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