MODPROD

Center for Model-Based Cyber-Physical Product Development

Tutorial 2

# FMI for Composite Modelling, Co-Simulation and Model Exchange

Andreas Heuermann

16th MODPROD Workshop, February 1-2, 2022



FH Bielefeld
University of
Applied Sciences

# Installation Instructions

FH Bielefeld
University of
Applied Sciences

# What you will need for this tutorial:

- OpenModelica
- Python3 installed with modules
  - OMPython
  - OMSimulator
- Jupyter Notebook

Note for Mac users: Use a virtual machine with Linux



FH Bielefeld
University of
Applied Sciences

# Documentation and Community Help

**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)

- GitHub github.com/OpenModelica/OMSimulator/

**Community**

- OpenModelica Forum openmodelica.org/forum
- Stack Overflow stackoverflow.com/
- Discord Modelica chatroom



FH Bielefeld
University of
Applied Sciences

# Installing OpenModelica

OpenModelica version v1.19.0-dev (greater version v1.17.0 is sufficient)

- OMSimulator is part of OMEdit
- GUI + CLI + scripting available
- Follow instructions for your platform
  Windows: https://openmodelica.org/download/download-windows
  Linux: https://openmodelica.org/download/download-linux


## Download Windows

| | | |
|---|---|---|
| **Official Release** | 1.18.1 (32bit/64bit) | • contains only validated new features<br>• intended for productive usage (commit history) (release notes) |
| **Stable Development** | 1.18.1 (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<br>• dev.betaxx versions are released in preparation to official releases for testing; no new features are added to beta versions, only bug fixes<br>• latest stable release: **1.18.1** (commit history) (release notes) |
| **Nightly Build** | 1.19.0-dev (32bit/64bit) | • built daily with the latest additions to the code base that passed the standard regression tests (commit history)<br>• intended to make the latest developments and enhancements available for testers and developers, not for productive usage<br>• features that are not subject to regression testing may get broken between one nightly build and the next |

# Installing Jupyter Notebook

Windows

- Install Anaconda
  Anaconda Individual Edition: anaconda.com/products/individual
- Start Jupyter Notebook (Anaconda3)

Linux

- Install Python 3 and pip3
- Install Jupyter Notebook

```
$ sudo apt update
$ sudo apt install python python3-pip
$ pip3 install jupyter
$ jupyter-notebook
```

# Installing Python Modules

Windows

- Install OMSimulator and OMPython modules
  Start Anaconda Prompt (anaconda3) and run:

```
> pip3 install OMSimulator
> echo %OPENMODELICAHOME%
C:\Program Files\OpenModelica1.19.0-dev-64bit\
> cd %OPENMODELICAHOME%\share\omc\scripts\PythonInterface
> python –m pip install –U .
```

# Installing Python Modules

Linux

- Install OMSimulator and OMPython modules
  Start a shell with Python 3 in PATH:

```
$ pip3 install OMSimulator
$ python –m pip install –U
https://github.com/OpenModelica/OMPython/archive/master.zip
```



FH Bielefeld
University of
Applied Sciences

# FMI and SSP Standards

# FMI and SPP Standards



# **F**unctional **M**ock-Up **I**nterface (FMI)

- Free standard
- Defines container and interface to exchange models
- Latest release: FMI 2.0.3
- Latest development build: FMI 3.0 (Beta)

fmi-standard.org

# **F**unctional **M**ock-Up **U**nit (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

# **S**ystem **S**tructure & **P**arameterization (SSP)

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

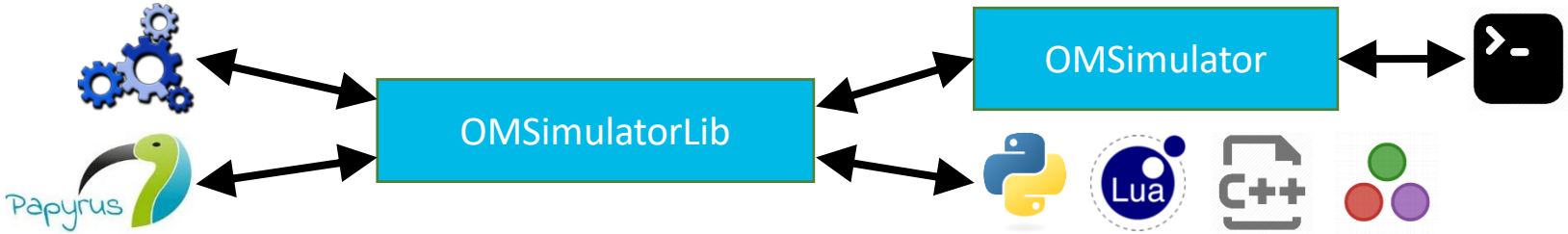
An overview

# What‘s new in OMSimulator

Current release: OMSimulator v2.1.1 (Jan 2021)

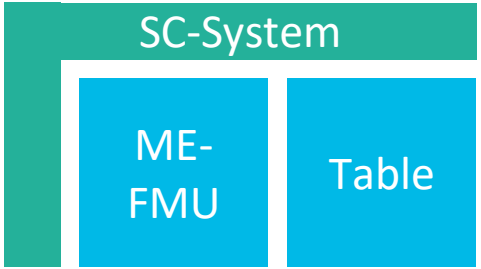
- SSP import/export
- Initialization of composite models
- Integration in OMEdit
- New Python API

# User Interface




- Command-line interface
- Scripting interface
- Graphical interface

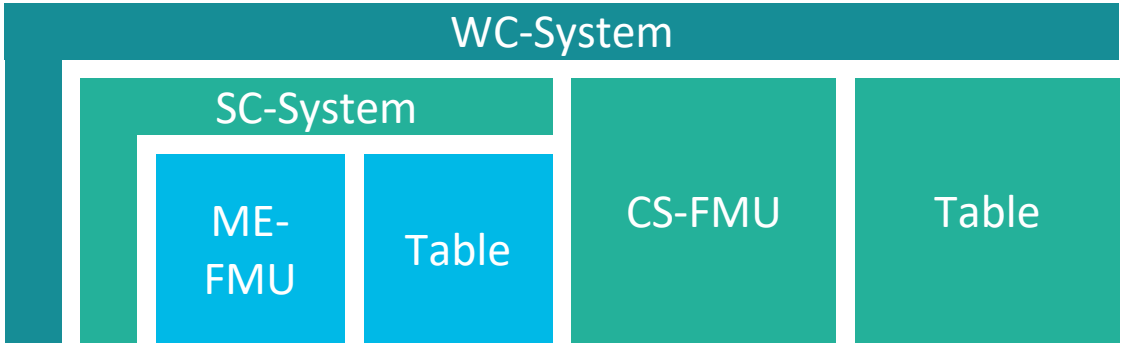
# Composite Model Structure - SC




**Strongly Connected System**

- Direct communication schema
- Detecting and handling algebraic loops
- Integration methods
  - Explicit Euler
  - SUNDIALS CVODE

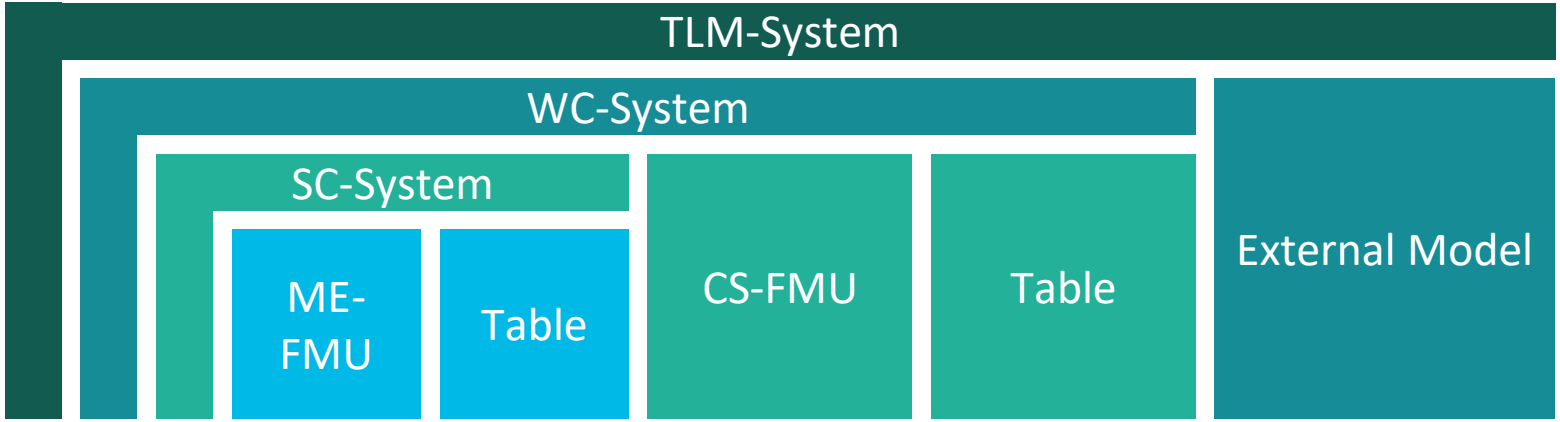
# Composite Model Structure - WC




**Weakly connected system**

- Communication at communication time points
- Extrapolation of inputs

# Composite Model Structure - TLM

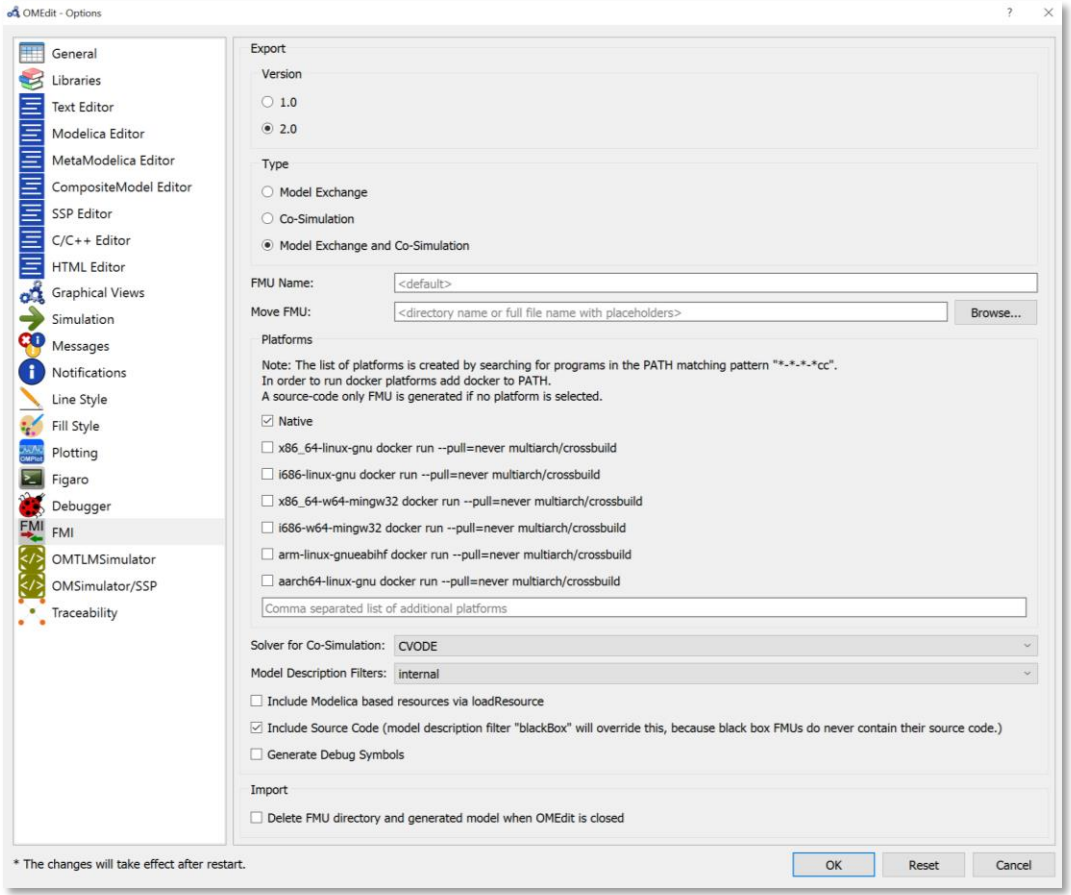



**Transmission Line Modelling**

- Physical signal connections

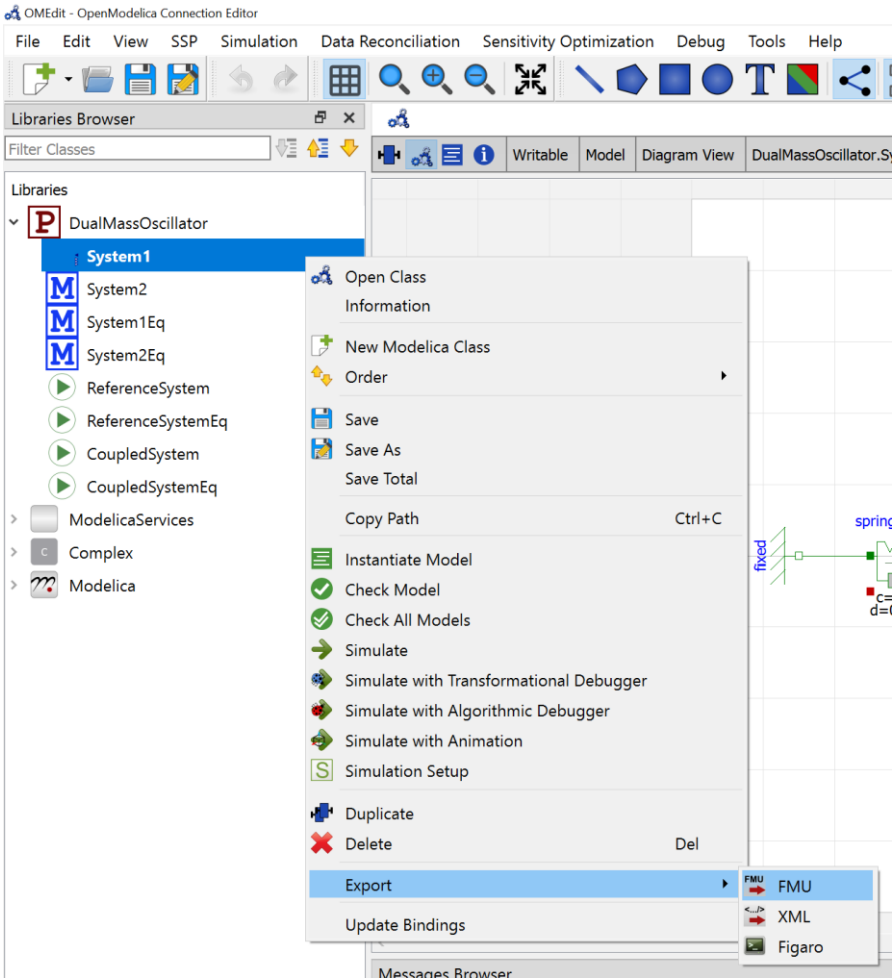
# FMI Export

# Export FMU from OpenModelica



# Export FMU from OpenModelica

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

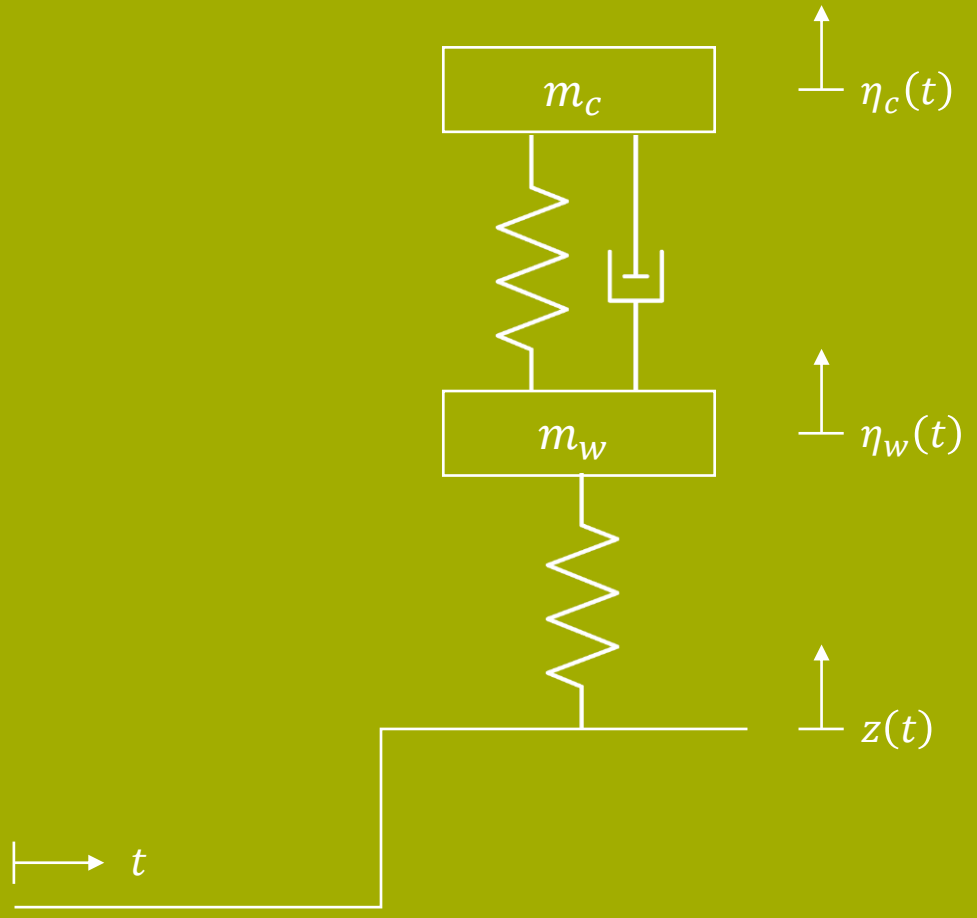





FH Bielefeld
University of
Applied Sciences

# Quarter Car Model

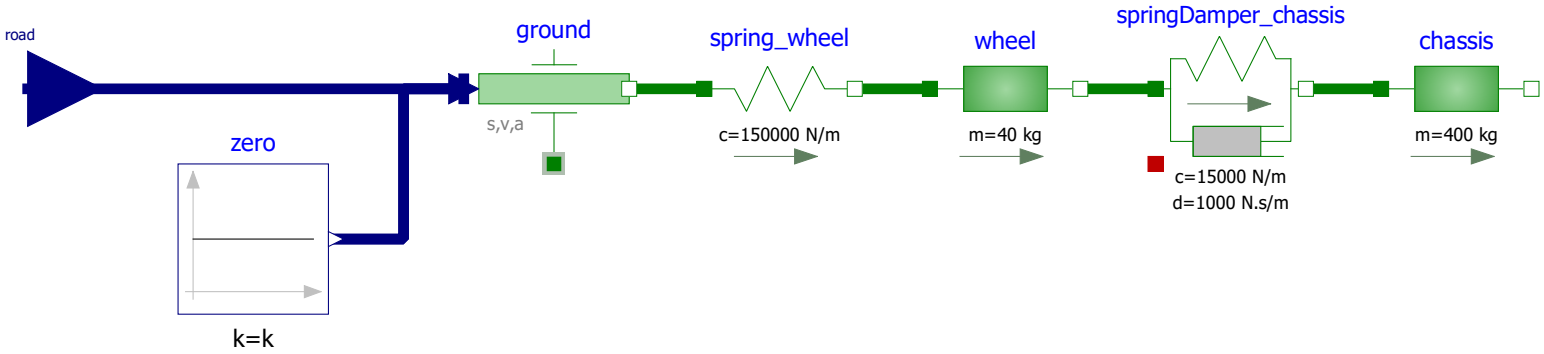
Exercise 1



# Exercise 1
# Quarter Car Model

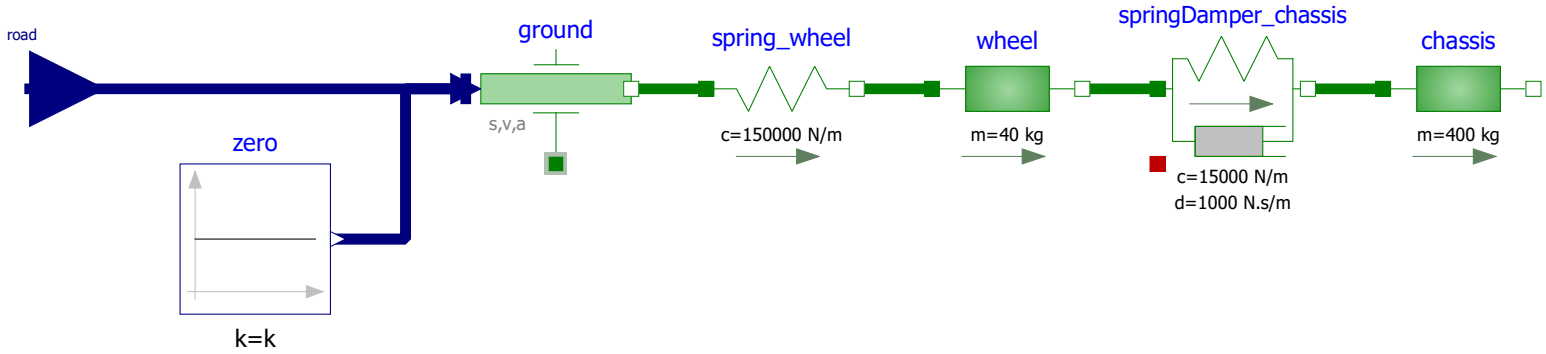
Task:

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



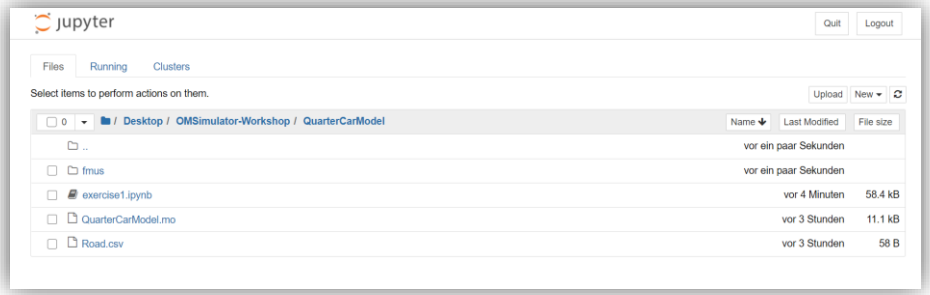
# Exercise 1
# Quarter Car Model

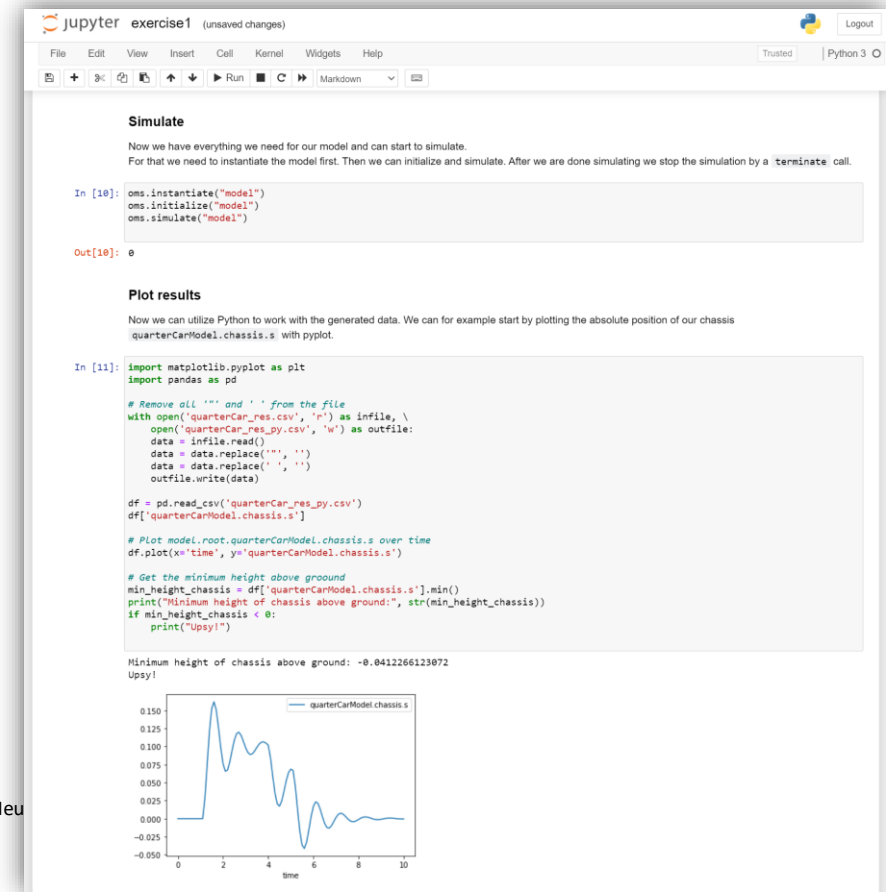
- Use Jupyter Notebook to open
  QuarterCarModel /exercise1.ipynb
  and start hacking!
- Install instructions can be found at the beginning of the presentation



# Exercise 1
# Quarter Car Model

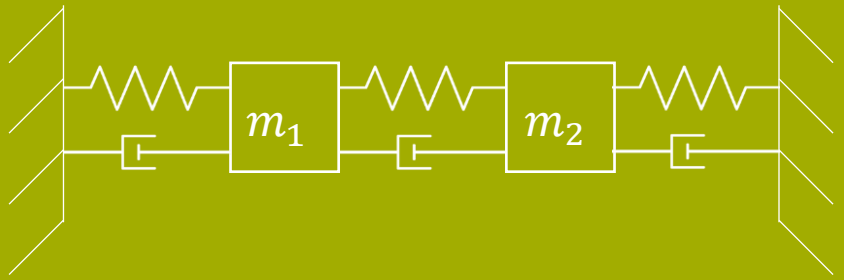
In Jupyter navigate to *exercise1.ipynb*



# Dual Mass Oscillator

Exercise 2



# Exercise 2
# Dual Mass Oscillator

Part I

Task:

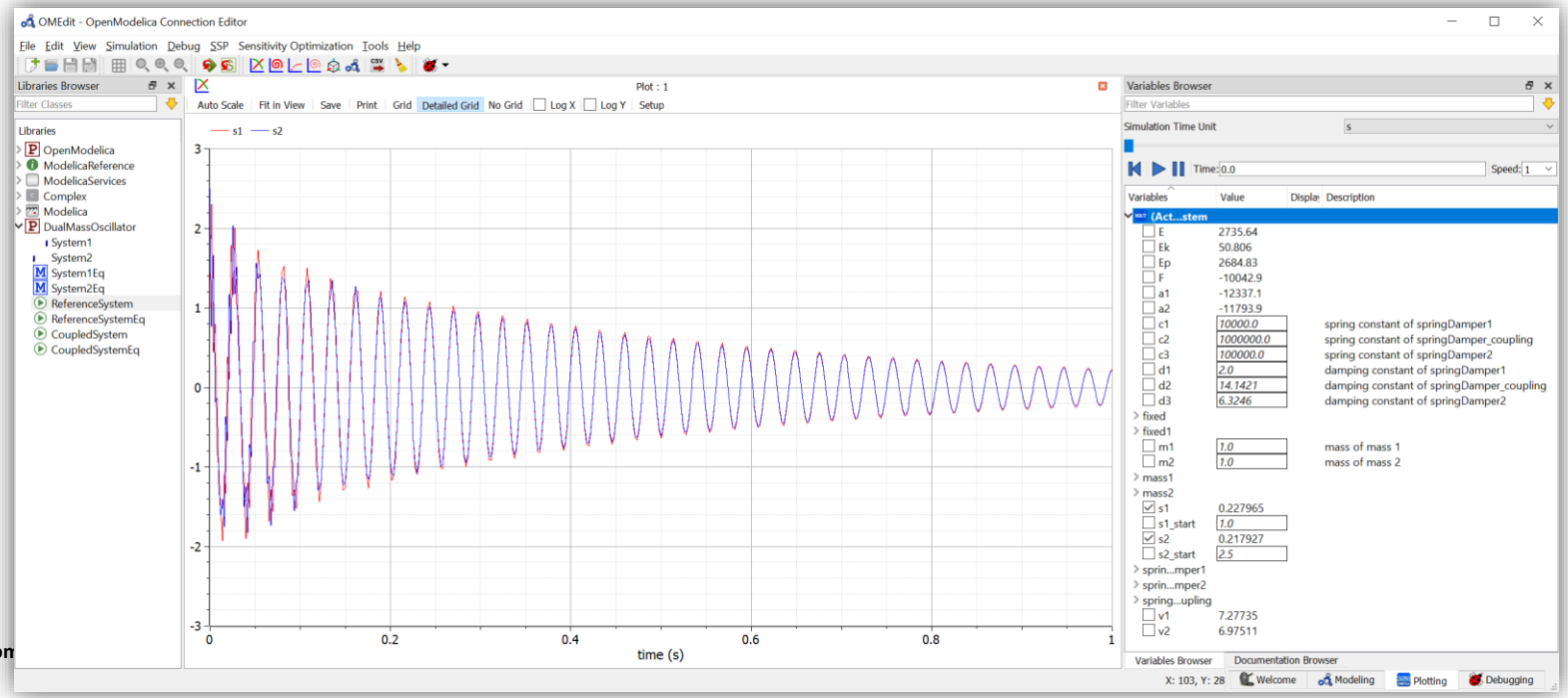
- Splitting the mechanical (reference) model into two subsystems using force-displacement coupling
- Defining interfaces for the FMUs
- Creating a FMU-based composite model (CS/ME)
- Set start values
- Simulate the composite model
- Export as SSP model



FH Bielefeld
University of
Applied Sciences

# Exercise 2
# Dual Mass Oscillator

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



# Exercise 2
# Dual Mass Oscillator

Part II

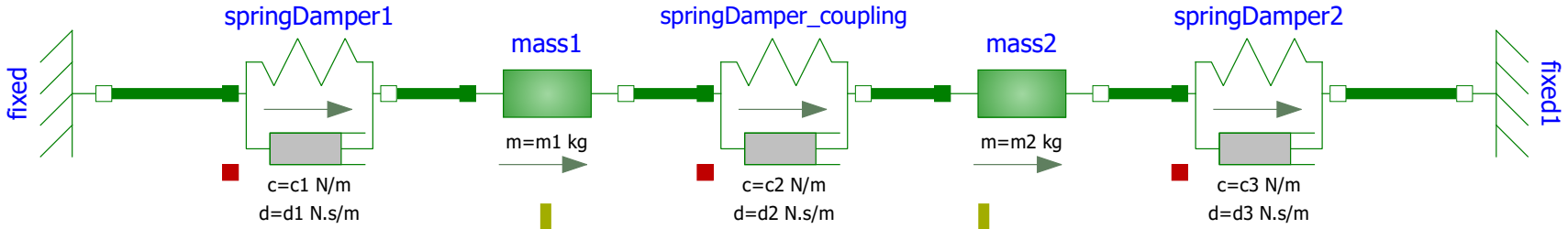
Task:

- 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`

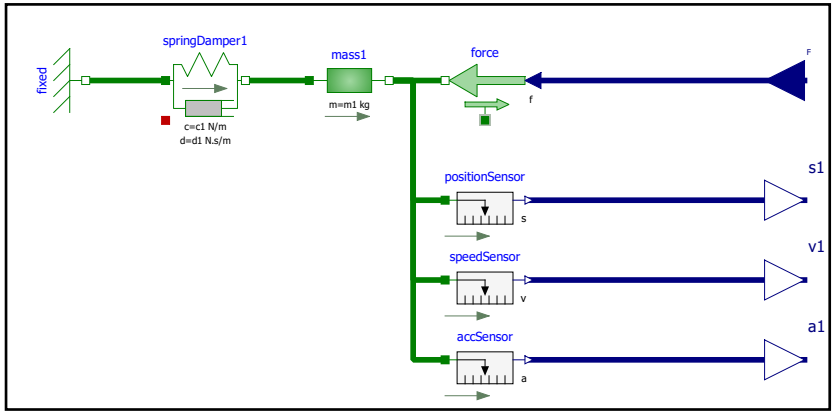


FH Bielefeld
University of
Applied Sciences

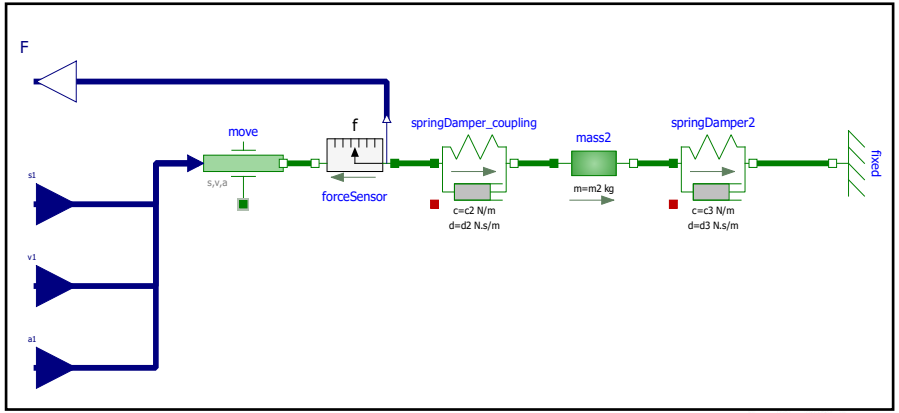
# Exercise 2
# Dual Mass Oscillator




## FMU1




## FMU2



# Exercise 2
# Dual Mass Oscillator

Part III

Task:

- 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



FH Bielefeld
University of
Applied Sciences

# Exercise 2
# Dual Mass Oscillator

Part IV

Task:

- 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

FH Bielefeld
University of
Applied Sciences