Numerically Robust FMI Co-simulation using Transmission Line Modelling with OMSimulator

Robert Braun, Linköping University

Robert Hällqvist, Saab AB

Dag Fritzson, AB SKF

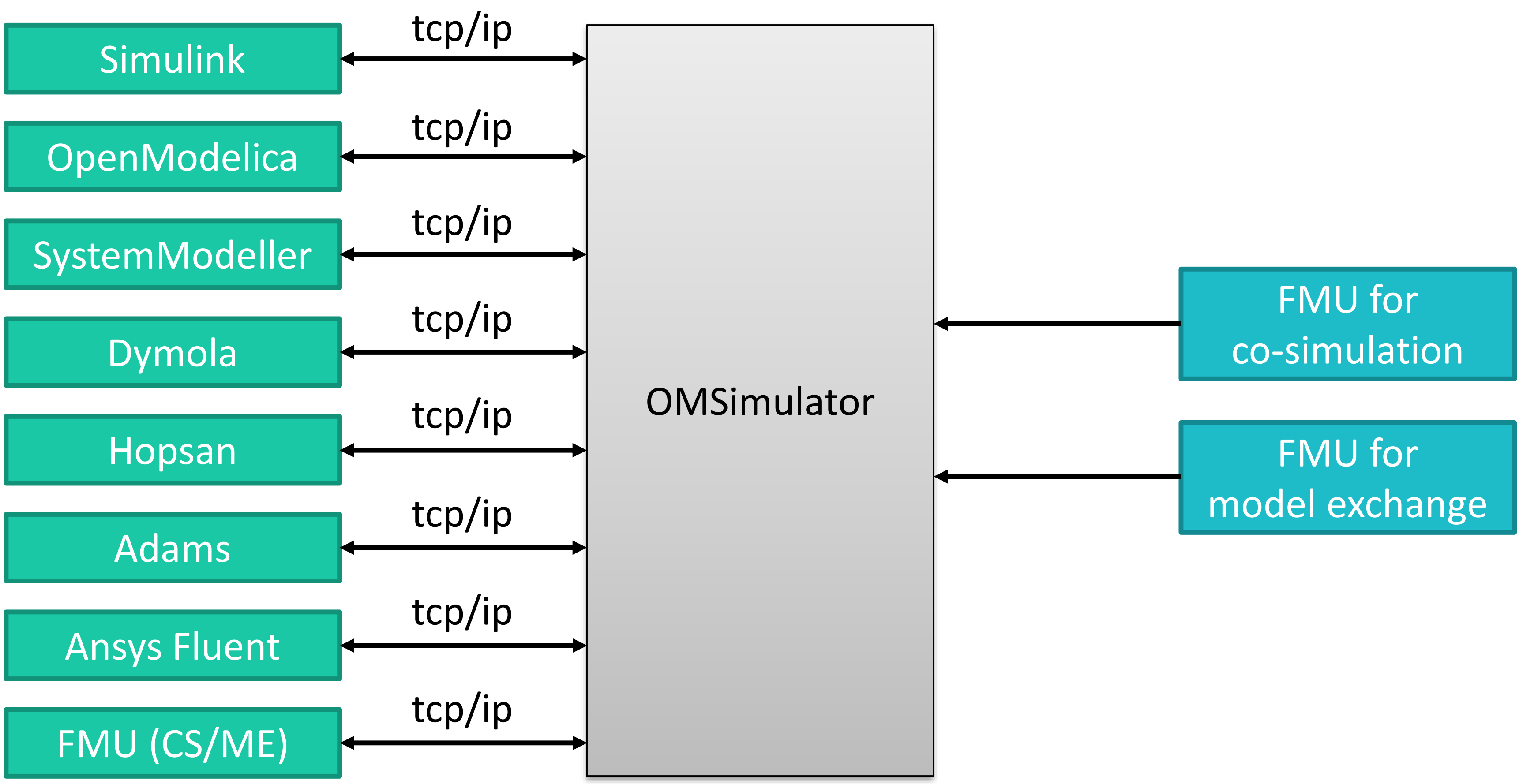
LINKÖPING UNIVERSITY

OMSimulator

- Key result from OpenCPS project
- Open Source (GitHub)
- Maintained by OSMC
- Features
  - Causal and acausal modelling
  - FMI import (ME/CS)
  - SSP support

openCPS

OMSimulator Overview



Socket Communication

- Busy-waiting synchronization
- External tool API:
  - getEffort: At any time during communiation step
  - setFlow: Whenever new output is produced

Background

Decoupling will delay certain variables:

Subsystem 1:
$$\begin{cases} \ddot{x}_1(t)M_1 + \dot{x}_1(t)B_1 = F_1(t) - F(t) \\ F(t) = k\big(x_2(t-T) - x_1(t)\big) \end{cases}$$

Subsystem 2:
$$\{\ddot{x}_2(t)M_2 + \dot{x}_2(t)B_2 = F(t-T) - F_1(t)$$

Hence, values from previous iteration must be used.

$\Rightarrow$ Reduced accuracy

$\Rightarrow$ Risk for instability

Background

~50 years ago, two research groups came up with the same idea:

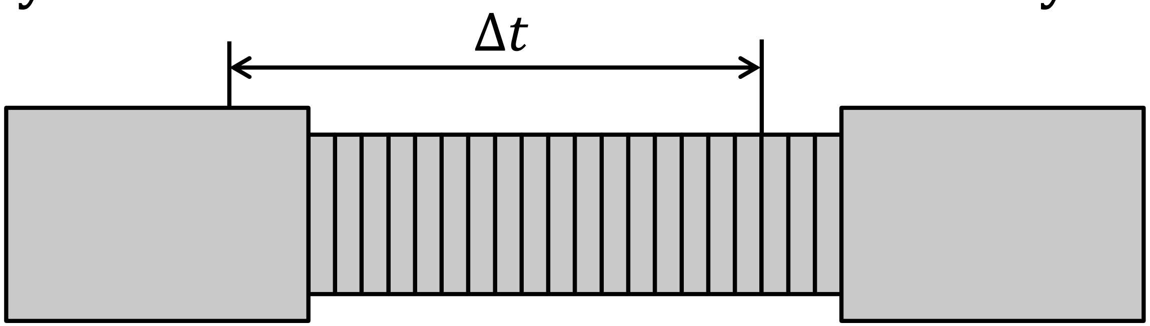
”In reality, information propagation speed is always limited by speed of sound (or light)”

- (Auslander, 1968) (Johns & O’Brian, 1971)

- Examples:
  - Hydraulic pipes
  - Mechanical springs,
  - Electrical transmission lines

Transmission Line Modelling (TLM)

Every physical element has a natural time delay:



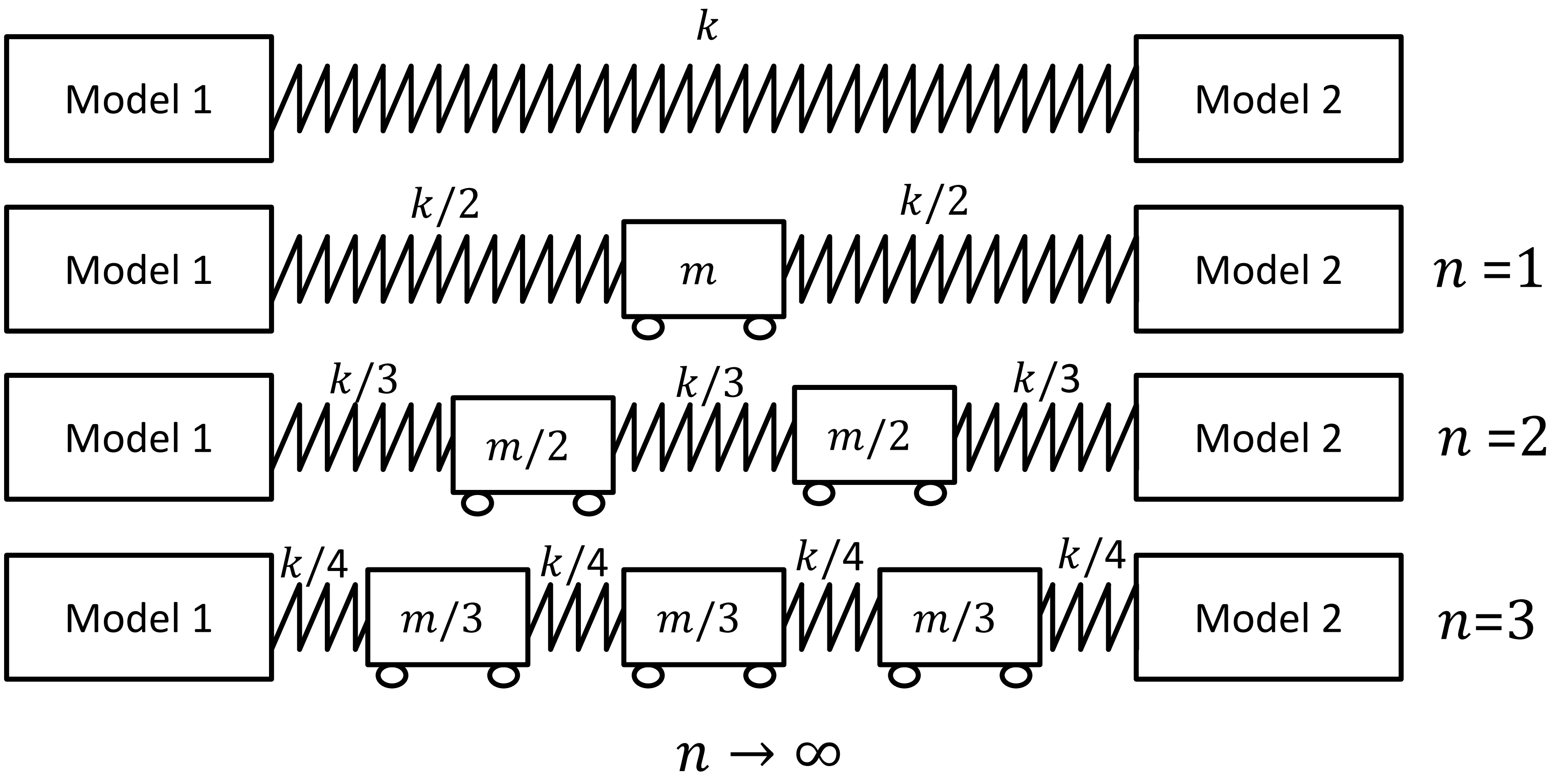
Physically motivated decoupling

→ numerical stability

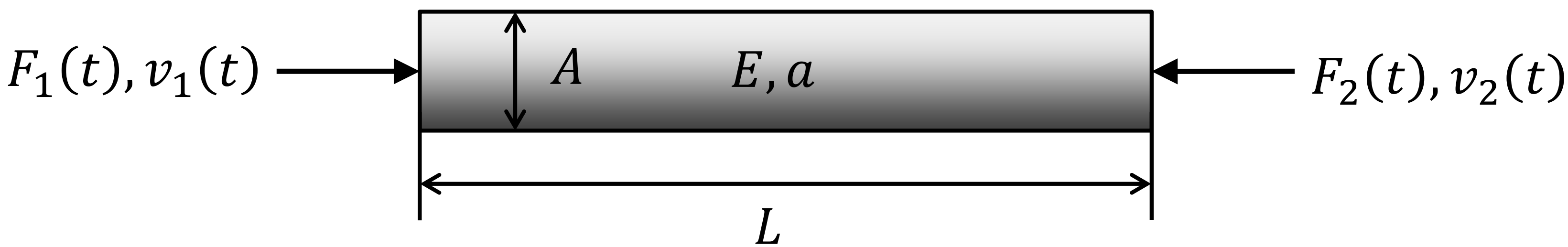
Asynchronous communication

→ independent time steps

Transmission Line Modelling (TLM)



Transmission Line Modelling (TLM)



TLM equations:

$$F_1(t) = F(t - \Delta t) + Z_c v_1(t) + Z_c v_2(t - \Delta t)$$
$$F_2(t) = F(t - \Delta t) + Z_c v_2(t) + Z_c v_2(t - \Delta t)$$

Capacitance (stiffness):

$$C = Z_c / \Delta t$$

Inductance (inertia):

$$L = Z_c \Delta t$$

Two-mass example again

Spring equation is replaced by the TLM equations:

$$\begin{cases} \ddot{x}_1(t)M_1 + \dot{x}_1(t)B_1 = F_1(t) - F_{s2}(t) \\ F_{s1}(t) = Z_C\dot{x}_1(t) + F_{s2}(t-T) + Z_C\dot{x}_2(t-T) \\ F_{s2}(t) = Z_C\dot{x}_2(t) + F_{s1}(t-T) + Z_C\dot{x}_1(t-T) \\ \ddot{x}_2(t)M_2 + \dot{x}_2(t)B_2 = F_{s2}(t) - F_1(t) \end{cases}$$

Two-mass example again

Splitting up the system now yields:

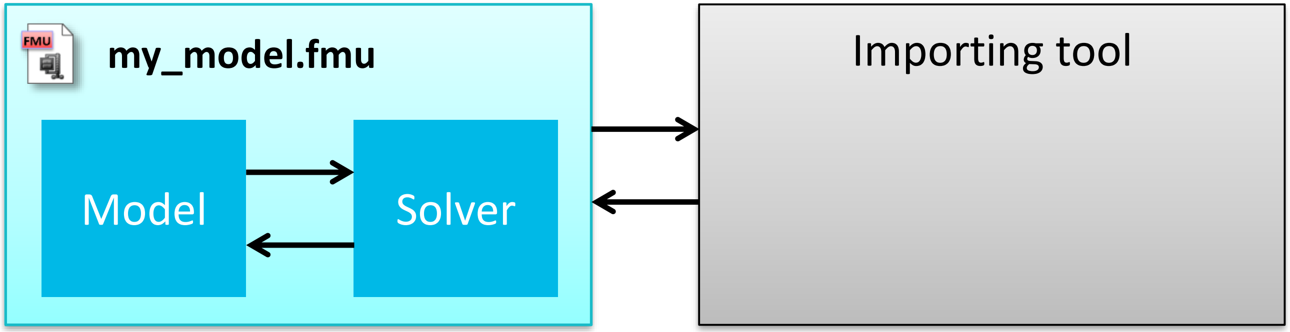
System 1: $\begin{cases} \ddot{x}_1(t)M_1 + \dot{x}_1(t)B_1 = F_1(t) - F(t) \\ F_{s1}(t) = Z_C\dot{x}_1(t) + F_{s2}(\mathrm{t} - \mathrm{T}) + Z_C\dot{x}_2(t - T) \end{cases}$

System 2: $\begin{cases} F_{s2}(t) = Z_C\dot{x}_2(t) + F_{s1}(t - T) + Z_C\dot{x}_1(t - T) \\ \ddot{x}_2(t)M_2 + \dot{x}_2(t)B_2 = F_{s2}(t) - F_2(t) \end{cases}$

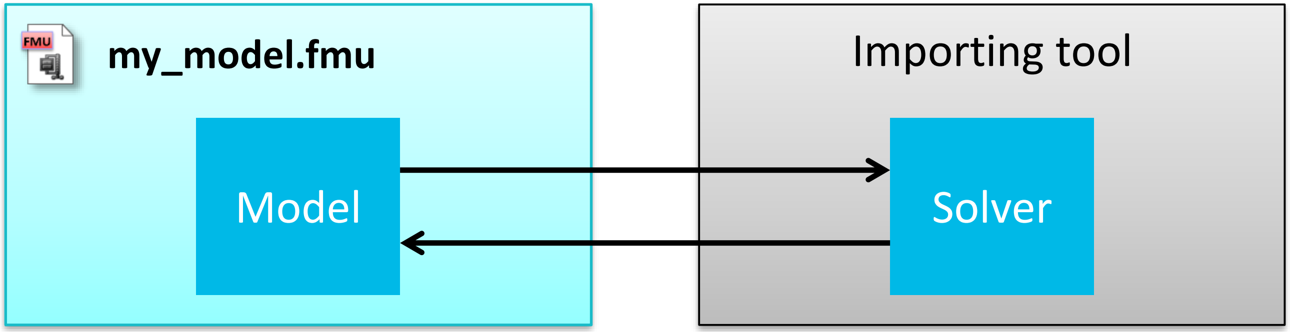
No *numerical* time delays!

Functional Mockup Interface (FMI)

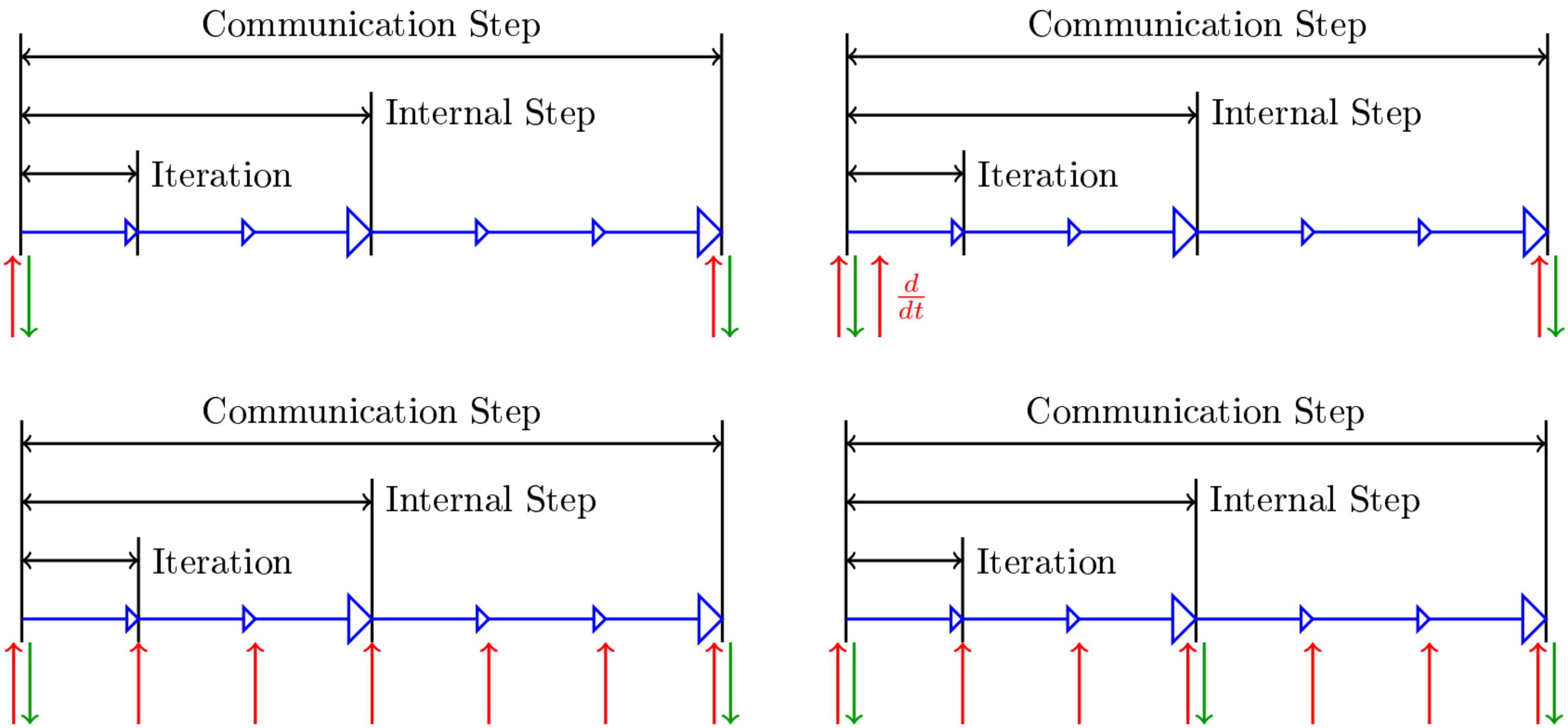
FMI for Co-simulation



FMI for Model Exchange

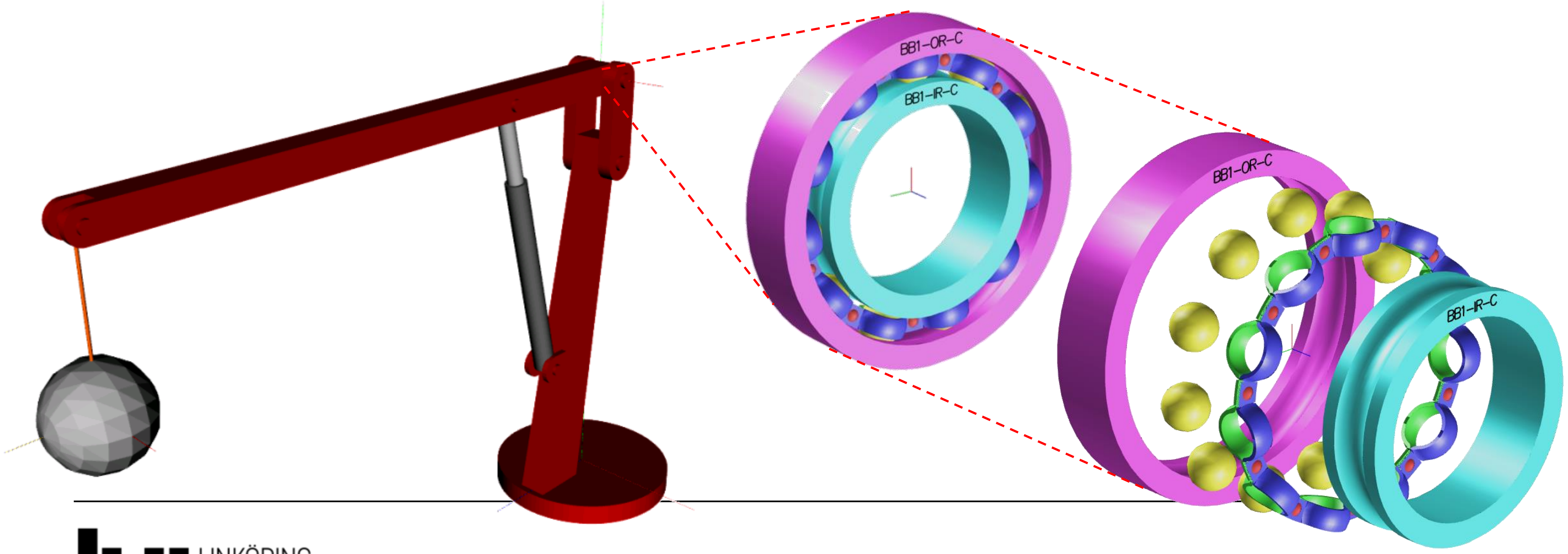


Communication Patterns

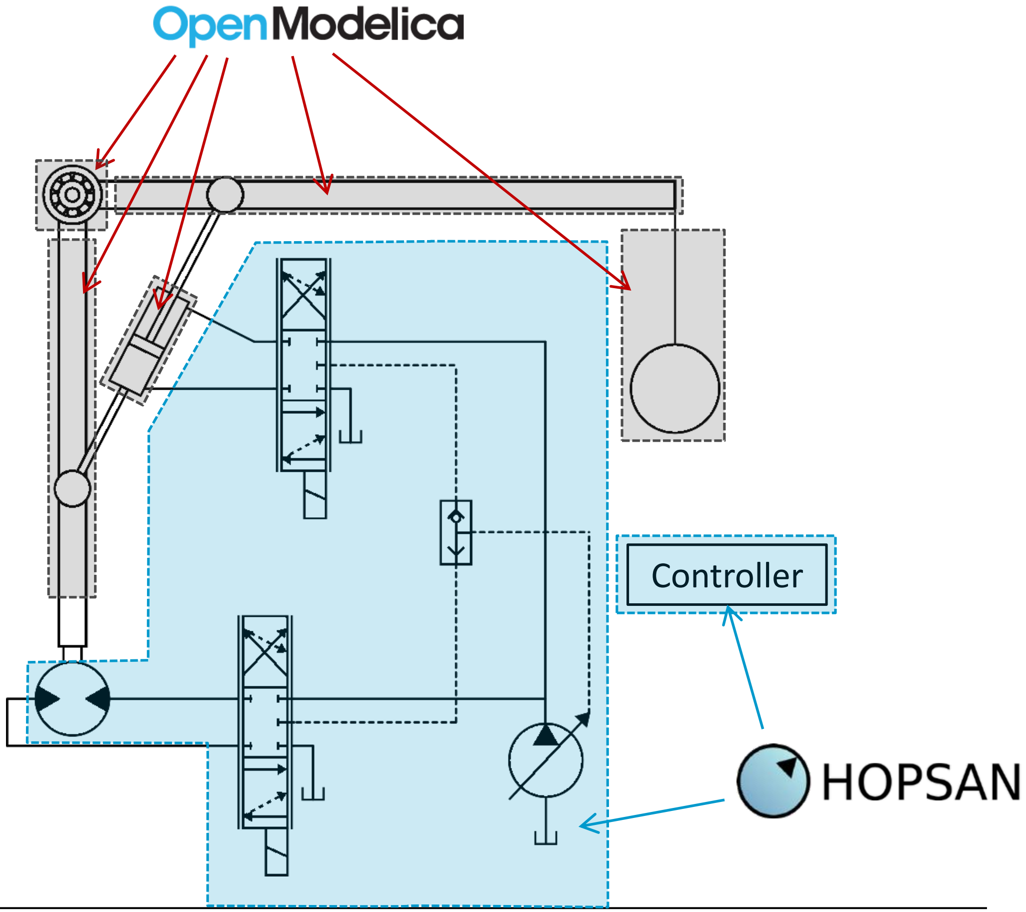


Demonstrator Model

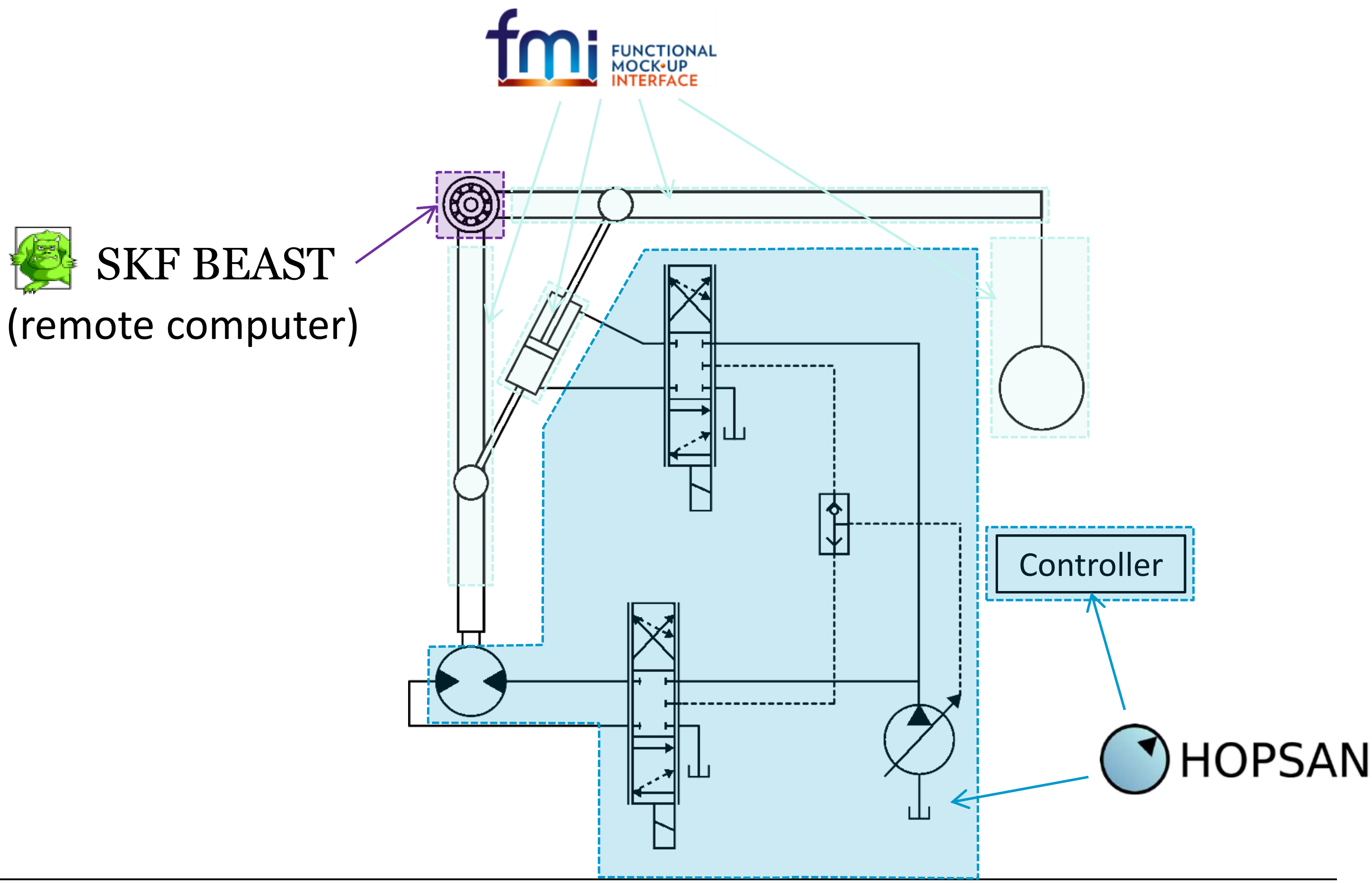
- Bearing in context for accurate boundary conditions
- System model typically supplied by customer/supplier
- Demonstrator model: Hydraulic crane



Demonstrator Model

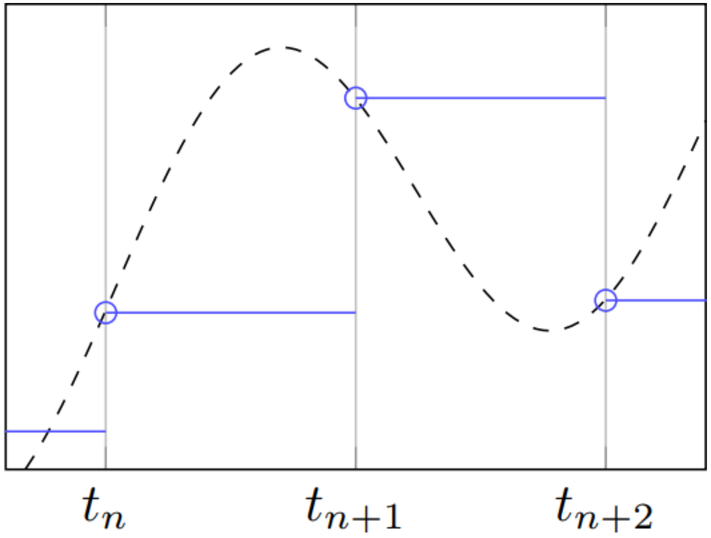


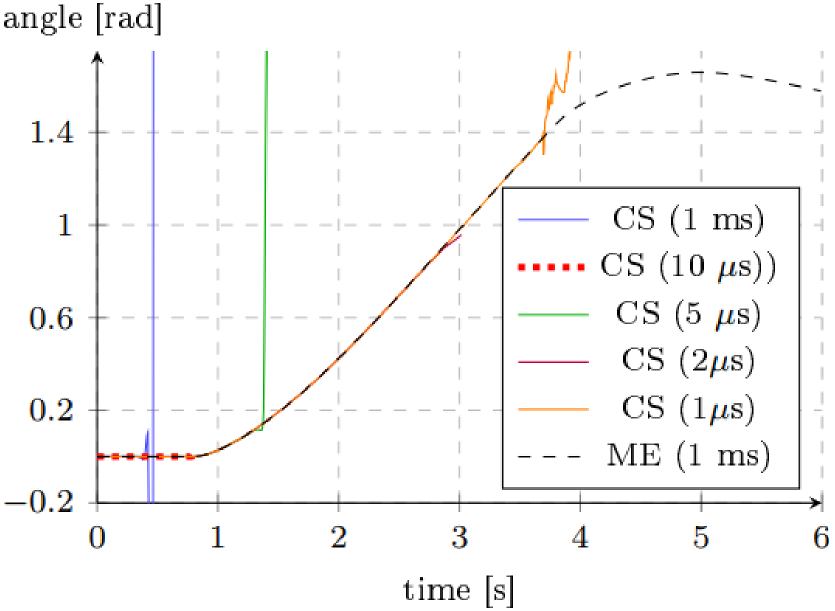
Demonstrator Model



Stability Analysis: Zero-order Hold

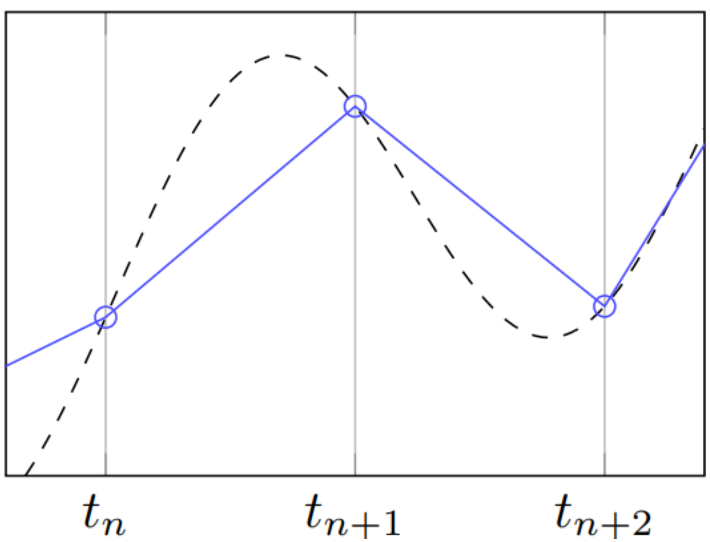
- Not stable after reducing step-size 1000 times!
- Compatible with all exporting tools

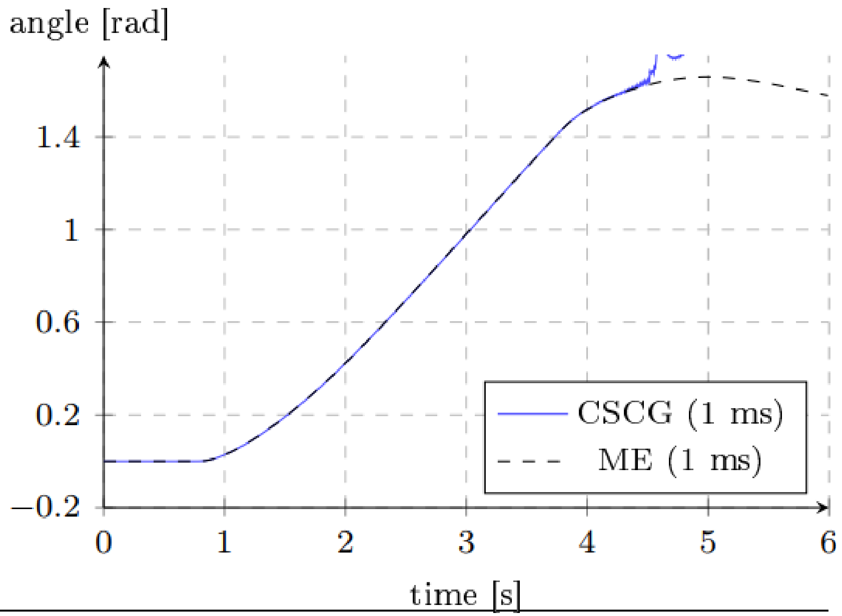




Stability Analysis: Input Derivatives

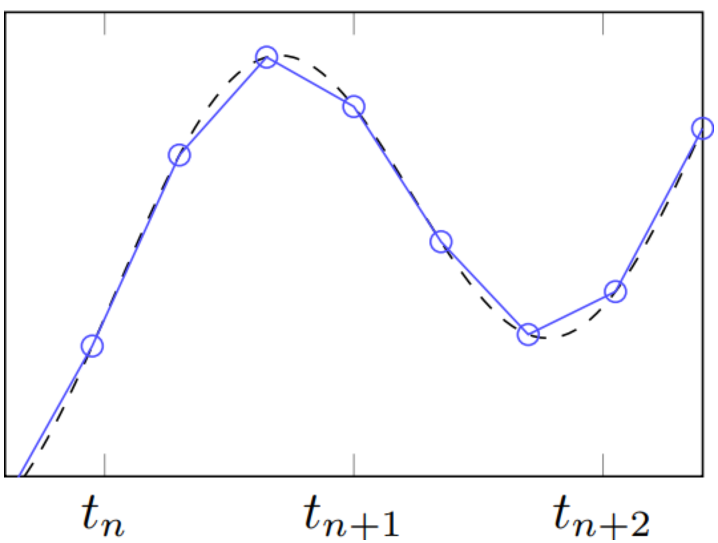
- Almost stable without reducing step-size
- Not supported by all exporting tools

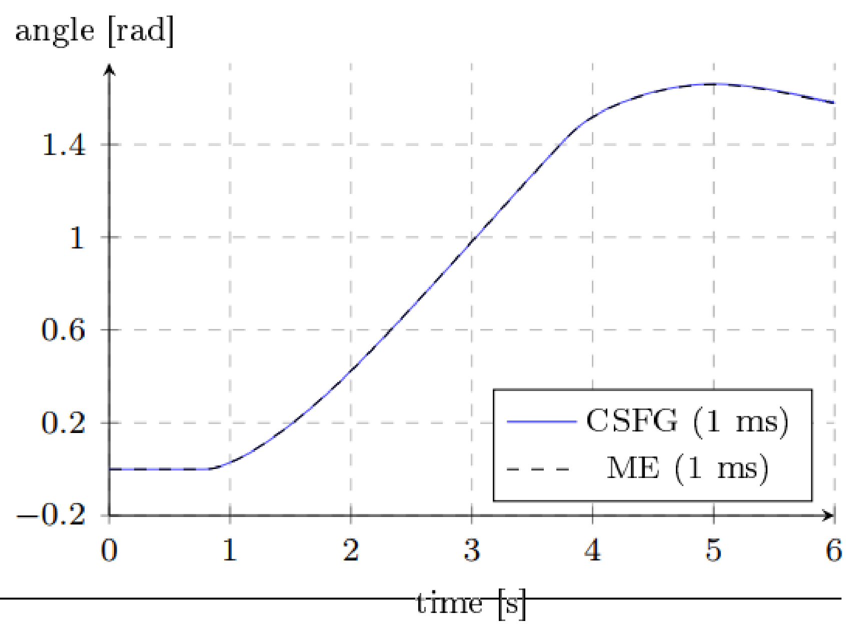




Stability Analysis: Fine-grained Interpolation

- Stable without reducing step-size
- Requires custom FMUs



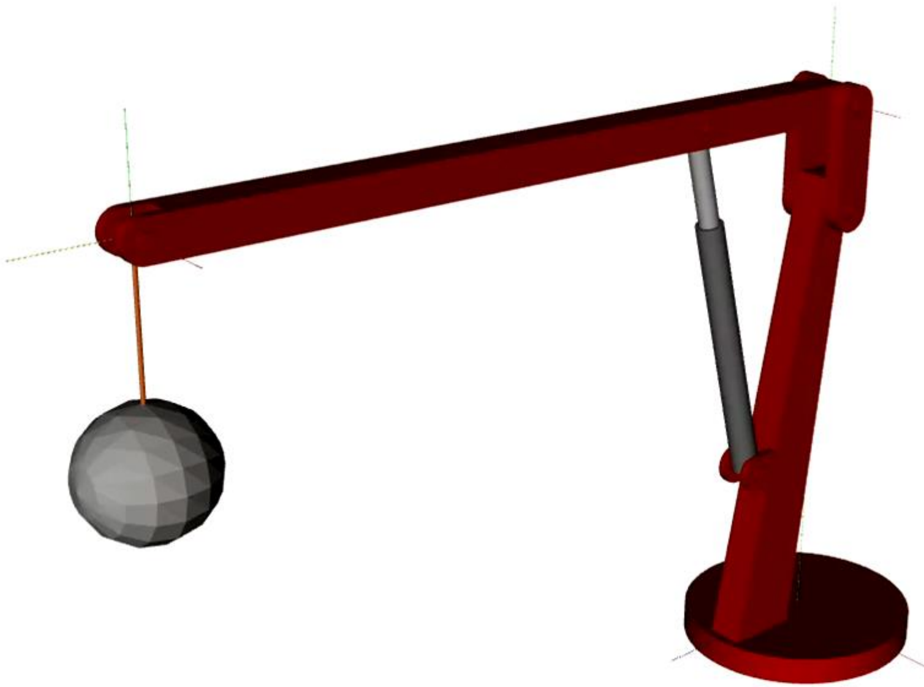


FMI Change Proposal (submitted)

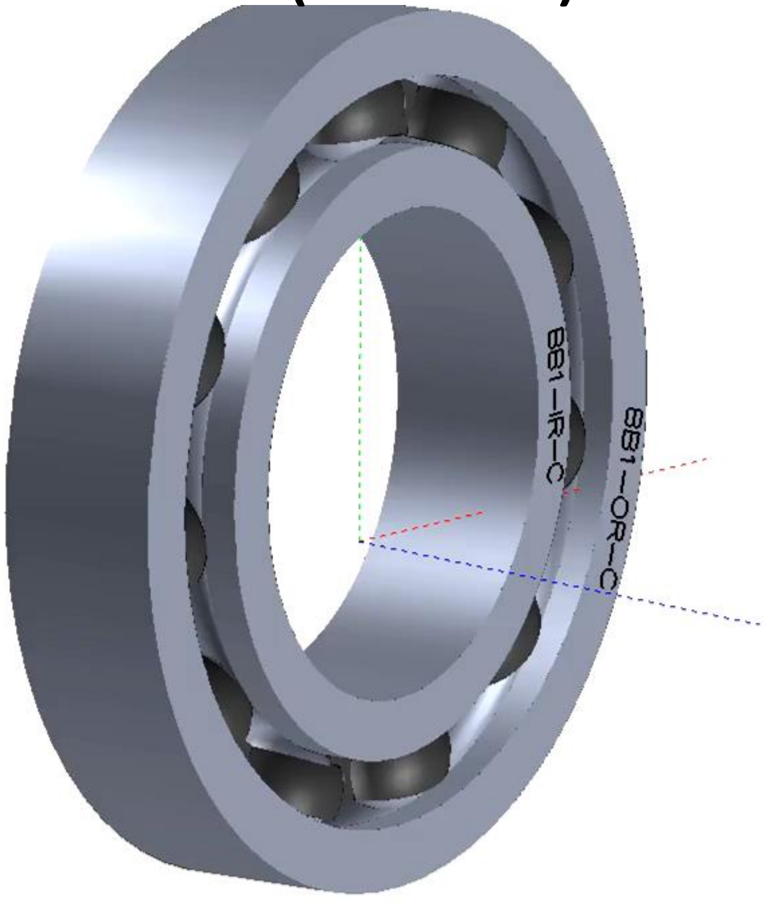
Callbacks for intermediate inputs/outputs

- Send output variables to master whenever produced
- Request input variables at any point during step

Simulation Results (OMSimulator)



Simulation Results (BEAST)



Robert Braun

www.liu.se

LINKÖPING UNIVERSITY