Webinar 2023-04-06

Co-simulation with LS-DYNA using FMI

Marcus Gustavsson - DYNAmore Nordic, an Ansys Company
marcus.gustavsson@ansys.com

Document information

Doc. no.: N/a

Revision: 1

Prepared for: Webinar

Project no.: N/a

Approved by: MG

Release date: 2023-04-06

Distribution: Approved for public release

DYNAmore Nordic AB
Brigadgatan 5
SE-587 58 LINKÖPING
Sweden

Org. no. 556819-8997
EC VAT:
SE556819899701

Phone: +46 (0)13 236680
Fax: +46 (0)13 214104
E-mail: info@dynamore.se
Web: www.dynamore.se

Co-simulation with LS-DYNA using FMI

An Introduction of Co-simulation with LS-DYNA through FMI

Introduction to FMI

Co-simulation using FMI in LS-DYNA

- LS-DYNA FMU Manager
- Remote
- *KEYWORD
- General comments
- Examples
- Python & FMI

Summary

Introduction to FMI

Functional Mock-up Interface (FMI) is a free and tool-independent standard that provides an interface for exchanging dynamic models between different software; maintained as by the Modelica Association.

FMI makes it possible to co-simulation between LS-DYNA and the 170+ tools that support the standard, https://fmi-standard.org/tools/, extending the capabilities of LS-DYNA.

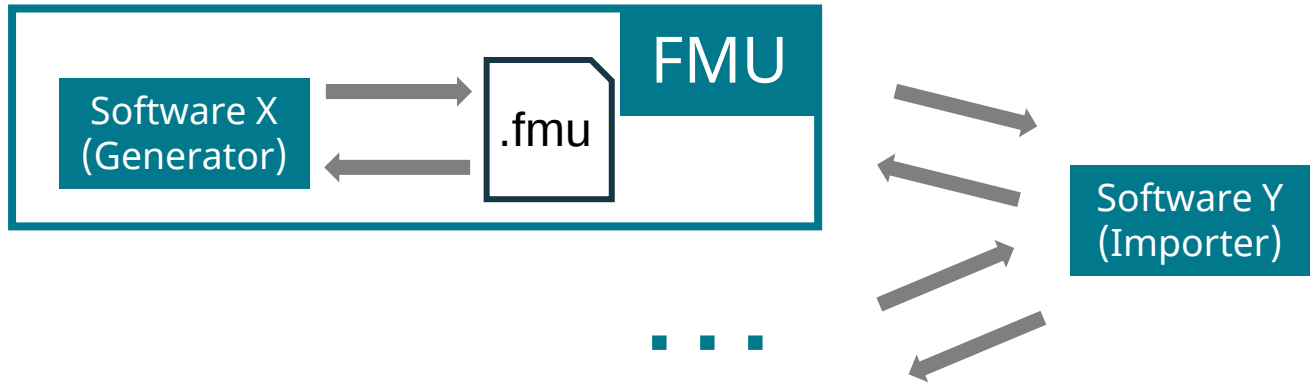
- Functional Mock-up Interface (FMI)
  - A tool independent standard to support model exchange and co-simulation between different software.
  - FMI 1.0 was released in 2010, FMI 2.0 in 2014, FMI 3.0 in 2022 (Currently not supported by LS-DYNA)
  - Modelica Association FMI Project: https://fmi-standard.org/
- Three Interface types
  - Model exchange
    - One software makes a model available to another software.
  - **Co-simulation - Only option for LS-DYNA**
    - Variables/solution data are exchanged using discrete communication time steps.
  - Scheduled Execution (from FMI 3.0)
    - Allows activation of individual model partitions by an external scheduler.
- A component that implements the FMI standard is called Functional Mockup Unit (FMU).

Introduction to FMI

Co-simulation

In co-simulation using FMI, data are exchanged via an FMU

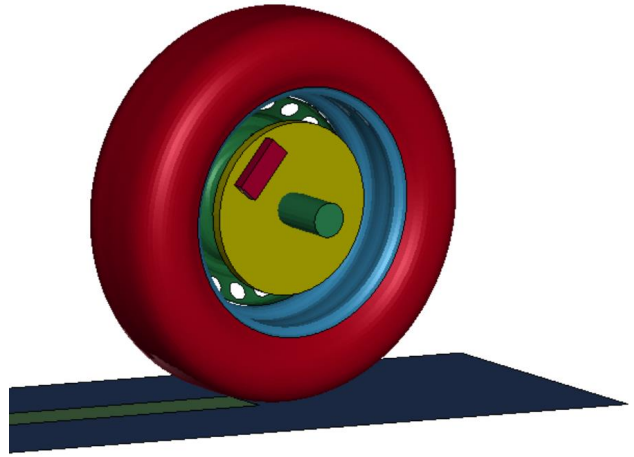
- FMU (Functional Mock-up Unit) is a ZIP archive (.fmu) serving as an API for data exchange between software
  - An FMU is generated by one software and then imported in another, the "importer"
  - The FMU typically acts as a container for its own solver in co-simulation, e.g., LS-DYNA
- Importer
  - One importer can import multiple FMU:s
  - The importer is responsible for
    - advancing the overall simulation time
    - exchange input and output data

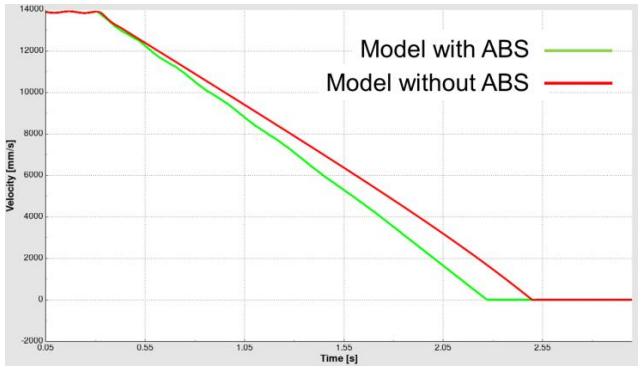


- For co-simulation, data exchange occurs at discrete communication time steps
  - The FMU and importer are solved independently in-between communication steps
  - Postprocessing is done for each software independently

Co-simulation using FMI in LS-DYNA

- LS-DYNA can both generate and import FMU
  - As FMU (Secondary)
    1. LS-DYNA is run in “FMU generation mode”
    2. LS-DYNA is run within the FMU and imported by another software
       - The other software’s co-sim. algorithm governs the FMU(ls-dyna)
  - As Importer (Primary)
    1. FMU is imported (FMU generated by other software)
       - Currently support for importing one FMU
    - LS-DYNA’s co-sim. algorithm governs the FMU
- Currently only support for FMI 1.0 & 2.0
- The co-simulation FMU and importer can
  - be local or remote
  - run in different environments, e.g., linux and windows
- Generate FMU and run co-sim. using the same version of LS-DYNA

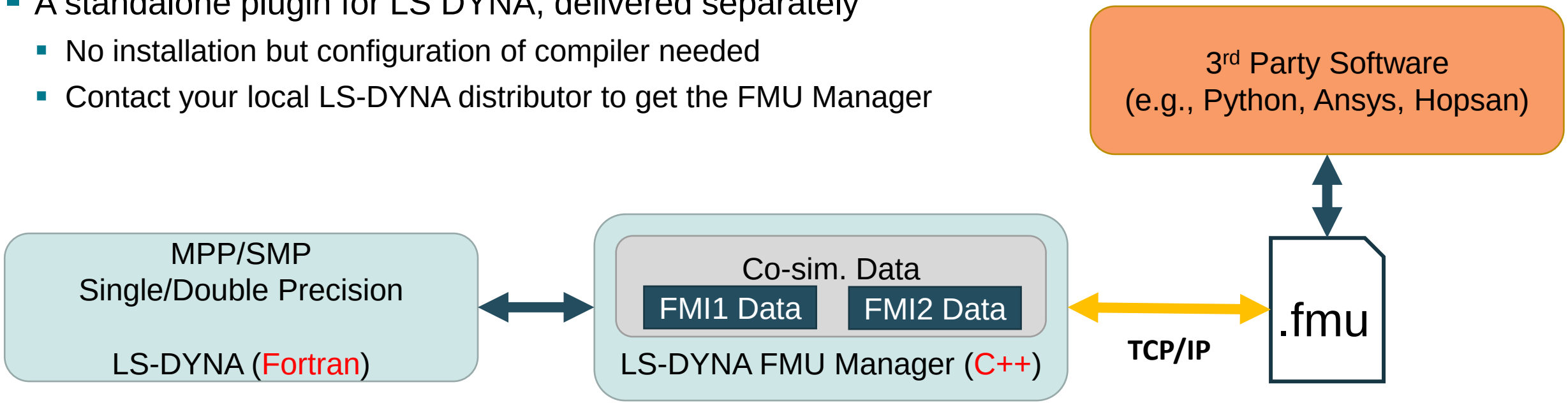




Co-simulation using FMI in LS-DYNA

LS-DYNA FMU Manager

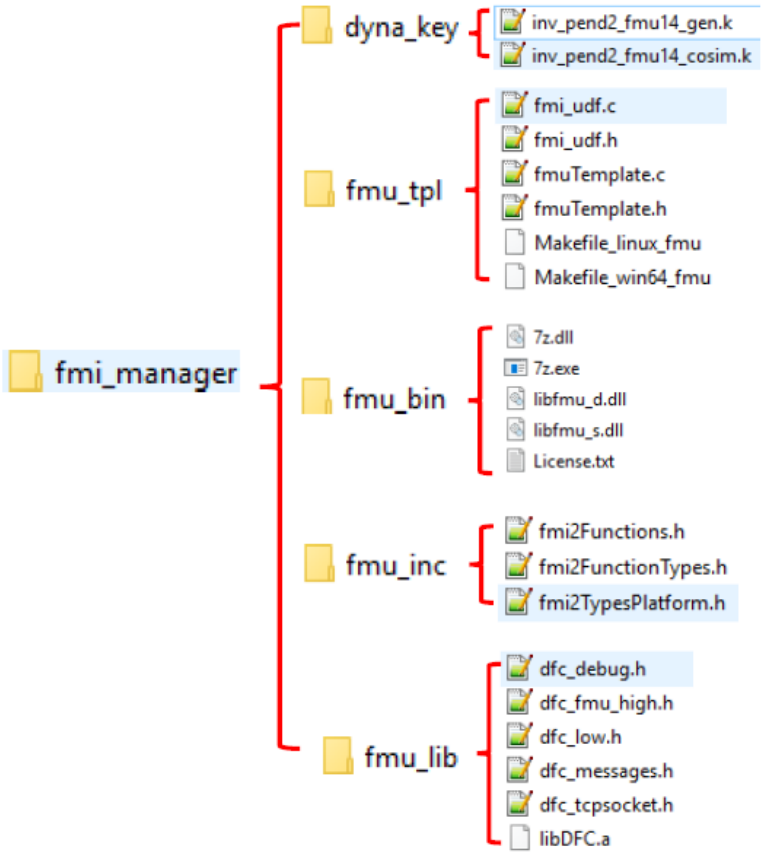
- LS-DYNA's FMI combability is realized through the LS-DYNA FMU Manager
  - The FMU Manager serves as an interconnect between LS-DYNA and FMU
  - Based on C++, LS-DYNA is based on Fortran and FMI on C
  - A standalone plugin for LS DYNA, delivered separately
    - No installation but configuration of compiler needed
    - Contact your local LS-DYNA distributor to get the FMU Manager



Co-simulation using FMI in LS-DYNA

LS-DYNA FMU Manager

- Usually, no need to modify any code in the manager, exceptions:
  - Generate FMU with LS-DYNA and import it back into LS-DYNA.
  - Modify fmi_dll.cin fmi_manager/fmu_tpl.



Examples

Templates for FMU generation.
Edit fmi_udf.c if one want tot generate FMU
and import it back to LS-DYNA

Core dynamic libraries of FMI manager
*.so for Linux and *. dll for Windows, zip.exe to generate FMU.
“s.dll” is for single precision, “d.dll” is for double precision.

Header files for FMI definitions.

Header files and library for TCP/IP socket communication.

Co-simulation using FMI in LS-DYNA

Remote

- The co-simulation FMU and importer can be remote and run on different computers
  - Simulation on HPC clusters
  - run in different environments, e.g., linux and windows
- **`*COSIM_FMI_CONTROL`**
  - <tcp> ip=127.0.0.1,port=39400
    - Specify IP address and port of computer running LS-DYNA.
    - If LS-DYNA and other code run on the same computer, set IP=127.0.0.1
- Running co-simulation on HPC cluster
  - What if you do not have the IP address of the head node?
    1. Generate FMU with arbitrary IP, e.g., 127.0.0.1
    2. Import FMU into other code
    3. Copy FMU to HPC, run input file with LS-DYNA.
    4. LS-DYNA will write HPC IP to ip_addr.txt and wait for connection.
    5. In other code, change FMU IP address and run co-simulation.
- Running LS-DYNA and the other software on different computers with different OS
  - FMU must be consistent with the system it's imported on
    - Example: co-simulation of LS-DYNA on a Linux HPC and Python on Windows → generate LS-DYNA FMU in Windows.

Co-simulation using FMI in LS-DYNA

***KEYWORD**

***COSIM_FMI_CONTROL**

APPID	OPT	MODE	FMI			
SETTING						

- Controls FMU generation and FMU settings.
- APPID: FMU identification
- OPT
  - EQ.G: Generation mode. LS-DYNA will generate a new FMU
  - EQ.C: Co-simulation mode. LS-DYNA will co-simulate with an existing FMU
- MODE
  - EQ.P: LS-DYNA is importer/primary
  - EQ.S: LS-DYNA is FMU/secondary
- SETTING
  - FMU settings. More details on next slide

Note: `*COSIM` was `*COSIMULATION` in early releases

Co-simulation using FMI in LS-DYNA

***KEYWORD**

***COSIM_FMI_CONTROL**

- SETTING
  - General settings for the FMU
  - A setting can span multiple lines
  - Max 80 characters per line
- `<fmudir>` if not specified it generates/searches FMU in the directory its run in

```
*COSIM_FMI_CONTROL
$                appid
   inverted_pendulum          G          S
$    settings
<home> C:\Users\username\FMU\FMU_Manager_5.1_112322
<fmudir> C:\Users\username\FMU\FMU_tests\inverted_pendulum
<tcp> ip=127.0.0.1,port=39400
<socket> nconnect=20,tdelay=1000,recvtimeout=0
```

Setting Name	Description	OPT	MODE
<fmudir>	Specify FMU directory	C, G	P, S
<fmu_input>	Generate FMU with additional input variables	G	P
<fmu_output>	Generate FMU with additional output variables	G	P
<home>	Specify the direction of the FMU manager	C, G	P, S
<hopsan>	Generate FMU / co-simulate with Hopsan	C, G	S
<name>	Match variables in LS-DYNA and FMU	C, G	P, S
<message>	Specify how often to print out the co-sim messages		
<parameter>	Generate FMU with additional parameter variables	G	P
<socket>	Generate FMU with specified socket properties	G	S
<tcp>	Generate FMU with specified IP and bind LS-DYNA with specified port	C, G	S
<time>	Specify time step and duration of co-simulation	C	P, S

Co-simulation using FMI in LS-DYNA

***KEYWORD**

***COSIM_FMI_INTERFACE**

APPID							
IMPEXP	REGTYP	REGID	FIELD	VINIT	RATIO	CID	REF

- Define variables to be communicated between the solvers.
- APPID: FMU identification
- IMPEXP: Import/export flag
  - EQ.IMP: Variables to be imported into LS-DYNA
  - EQ.EXP: Variables to be exported from LS-DYNA

Co-simulation using FMI in LS-DYNA

***KEYWORD**

***COSIM_FMI_INTERFACE**

```
*COSIM_FMI_INTERFACE
$              appid    inverted_pendulu
$   impexp     regtyp      regid      fiel
$ output
        EXP       PART          1          D
        EXP       PART          1          V
        EXP       PART          2          A
        EXP       PART          2          W
        EXP       NODE       3976          D
        EXP       NODE       3976          D
        EXP       NODE       4870          D
        EXP       NODE       4870          D
$ input
        IMP       PART          1          F
```

- REGTYP: Type of interface region
  - EQ.NODE: Single node
  - EQ.NSET: Node set
  - EQ.SSET: Segment set
  - EQ.PART: Rigid part
  - EQ.FUNC: User defined curve function to be exported only.
  - EQ.CURV: User defined curve to be imported only.
  - EQ.SESW: Sense switch to be imported only.
- REGID: ID of the corresponding region type in LS-DYNA. (Negative ID imports th
- FIELD: Allowed fields on next slide.
- INIT: Initial value
- RATIO: Scale factor
- CID: Coordinate system ID

FIELD	Description	REGTYP	Export	Import	REF
CX, CY, CZ	Position	NODE, NSET, PART	✓		0, 1
DX, DY, DZ	Translational displacement in *X*, *Y*, and *Z*, respectively	NODE, NSET, PART	✓		0, 1, 2
VX, VY, VZ	Translational velocity in *X*, *Y*, and *Z*, respectively	NODE, NSET, PART	✓		0, 1, 2
ACCX, ACCY, ACCZ	Translational acceleration in *X*, *Y*, and *Z*, respectively	NODE, NSET, PART	✓		0, 1, 2
AX, AY, AZ	Angular displacement in *X*, *Y*, and *Z*, respectively	PART	✓		0, 1
WX, WY, WZ	Angular velocity in *X*, *Y*, and *Z*, respectively	PART	✓		0, 1
WDTX, WDTY, WDTZ	Angular acceleration in *X*, *Y*, and *Z*, respectively	PART	✓		0, 1
BAGP, BAGV, BAGT	Airbag pressure, volume and temperature	BAG	✓		N/A
FX, FY, FZ	Force in *X*, *Y*, and *Z*, respectively	NODE, NSET, PART		✓	0, 1
P	Pressure	SSET		✓	N/A
TX, TY, TZ	Torque	PART		✓	0, 1
Variable name in FMU	Defined curve function	FUNC	✓		N/A
Variable name in FMU	Defined curve	CURV		✓	N/A
SW1-SW4, SWA-SWD	Sense switch	SESW		✓	N/A

76
76
70
70

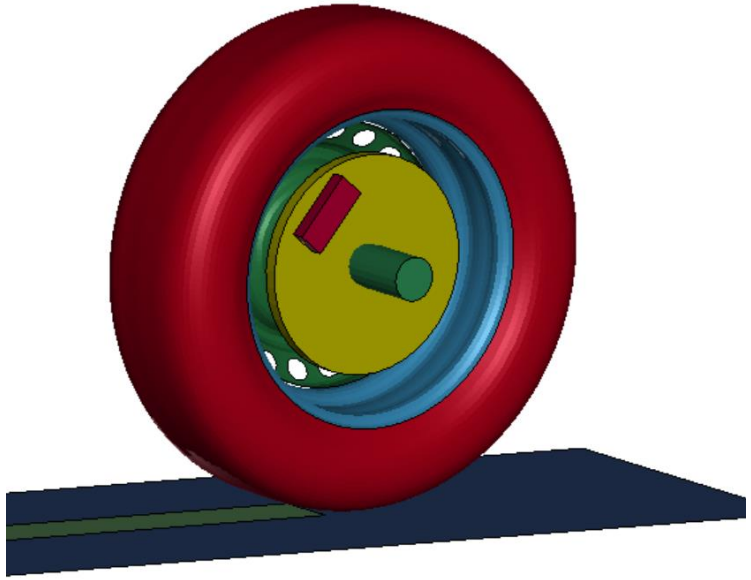
Co-simulation using FMI in LS-DYNA

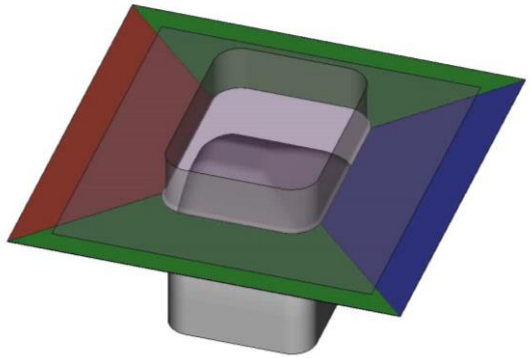
General comments

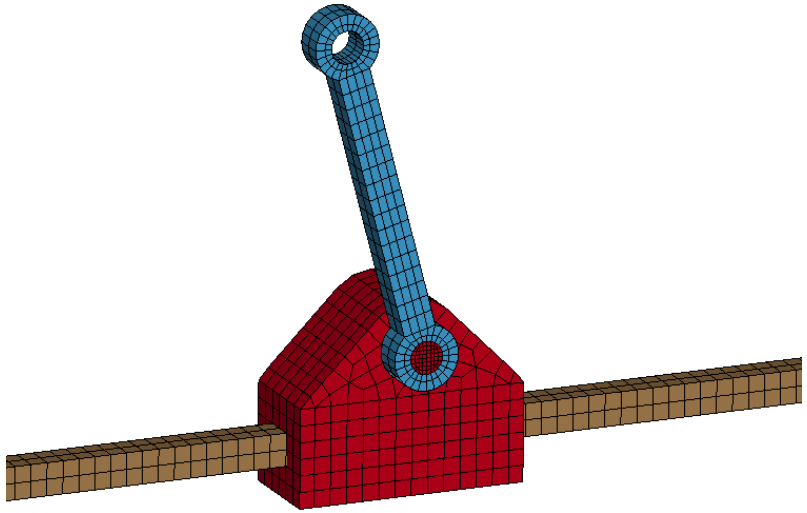
- Possible co-simulation setups
  - LS-DYNA <-> 3rd Party Software
  - LS-DYNA <-> FMU (.fmu by itself)
  - LS-DYNA <-> LS-DYNA
    - However, there are other options for specific applications, e.g., multiscale simulation of spotwelds
    - But it is possible to run two instances of LS-DYNA in co-simulation with FMI for a general application

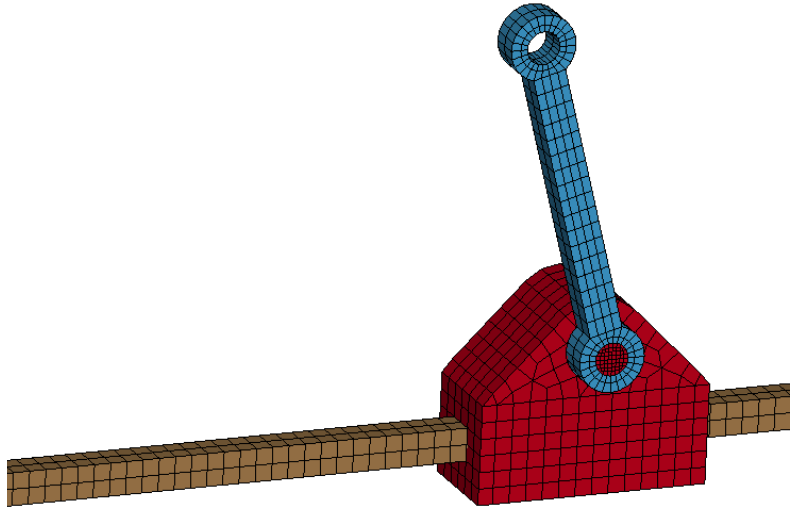
- When running LS-DYNA as FMU, make use of *CASE so that no modification is needed between generation of FMU and co-simulation
  - Case 1 contains **`*COSIM_FMI_CONTROL`** for generation
  - Case 2 contains **`*COSIM_FMI_CONTROL`** for co-simulation (running as secondary)

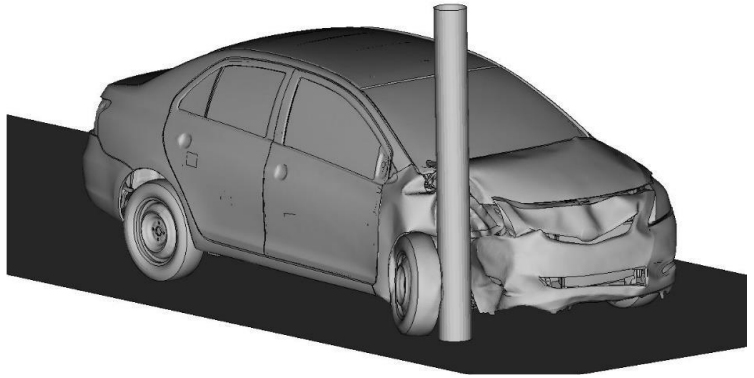
Examples

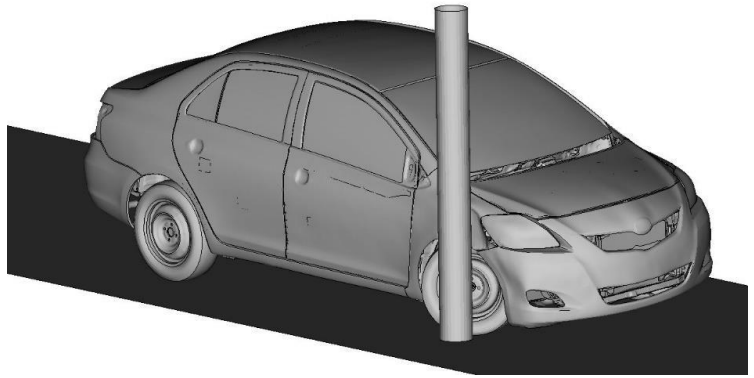






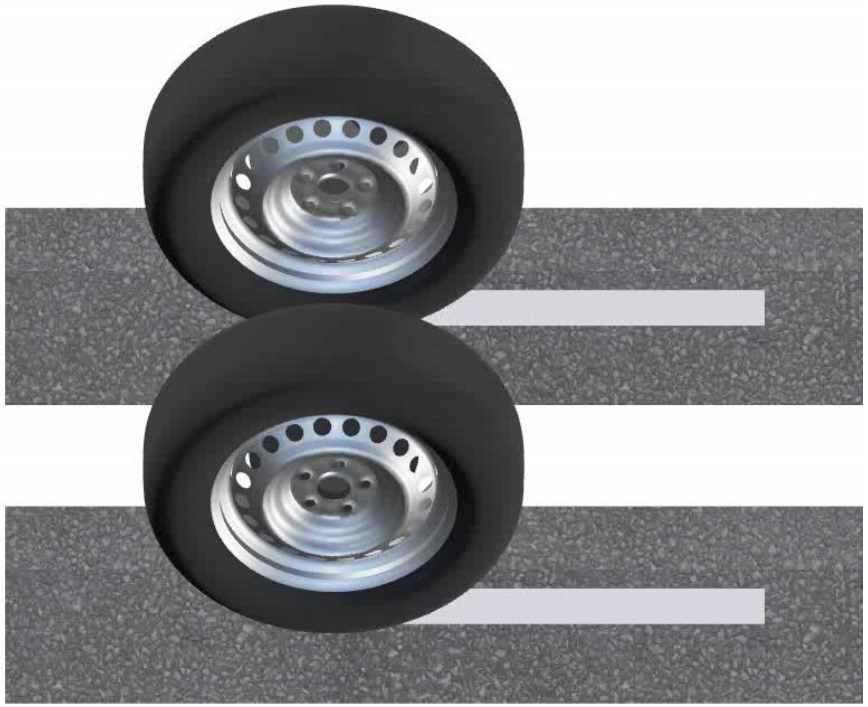






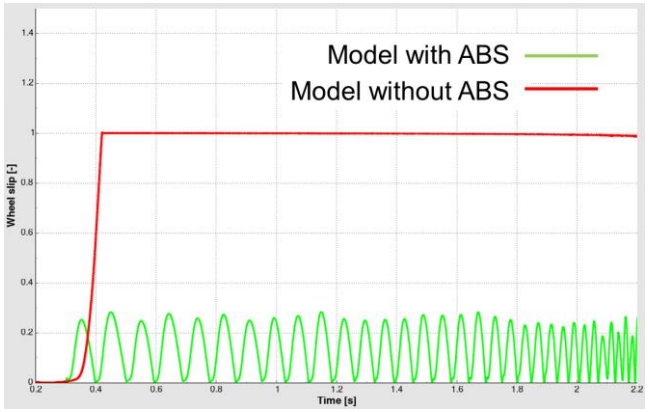
Examples

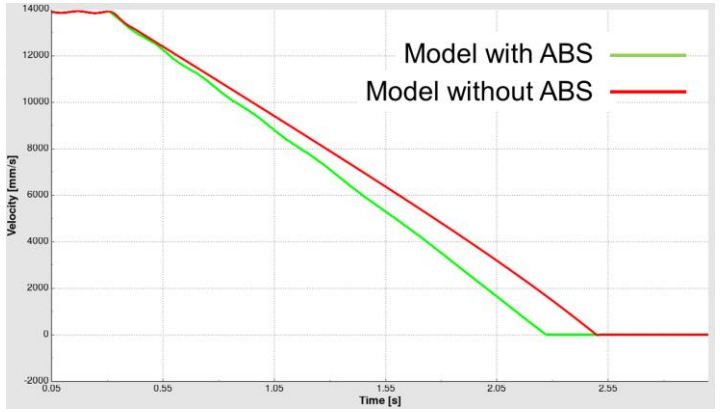
ABS

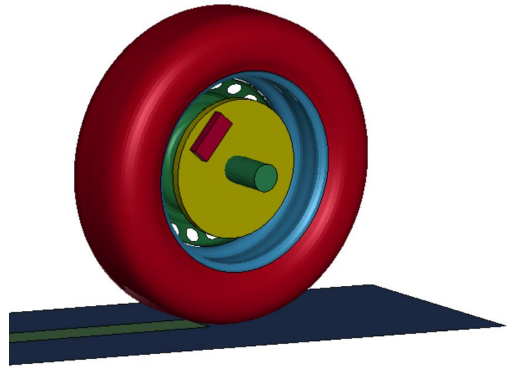


Model with ABS

Model without ABS

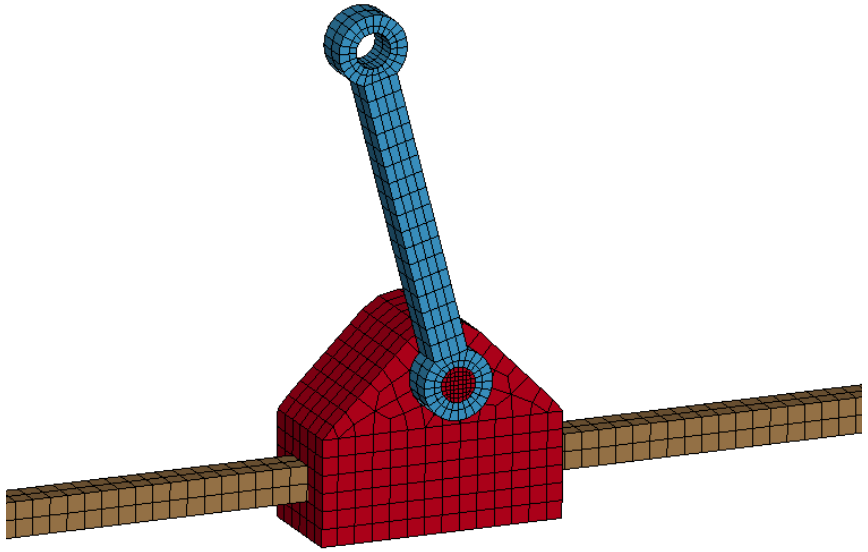




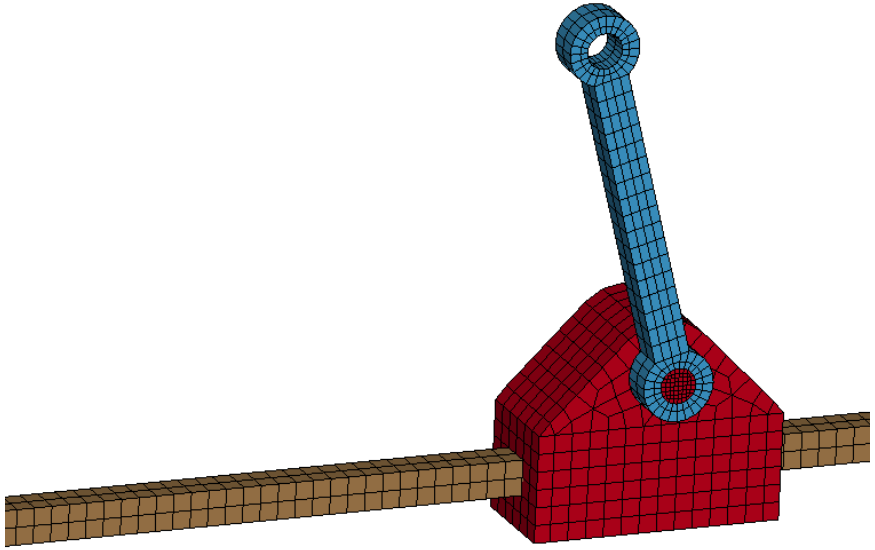


Examples

Inverted Pendulum with Controller in Python



without controller



with controller

Examples

Inverted Pendulum with Controller in Python

- See example files.

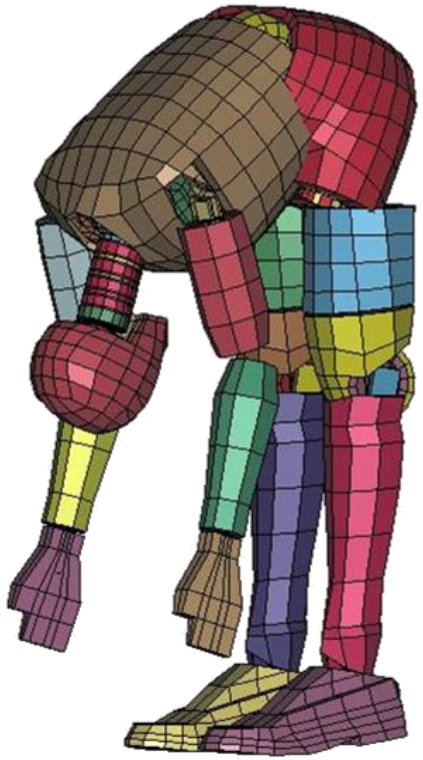
Python & FMI

- Two examples of packages that allow for the use for FMU:s in Python
  - FMPy
    - GitHub - FMPy
  - PyFMI
    - GitHub - PyFMI
- Note that syntax for communicating with the FMU is different, so python scripts aren’t completely interchangeable, some modification is required.

Summary

- FMI provides a general way to co-simulate LS-DYNA with 170+ other software
- Co-simulation in short: An FMU is generated by and contains a software and is imported to another software (importer). The importer controls time advance and data exchange, that occurs at discrete time steps; the two-software run independently in-between communication time steps.
- FMI can be used across different systems, e.g, linux and windows, but remember that the FMU must be generated in the environment that it will be imported in.
  - Use the same version of LS-DYNA for generation of FMU and co-simulation.
- Contact your local distributor to get the LS-DYNA FMU Manager

Thank You



DYNAMore Nordic AB
Brigadgatan 5
587 58 Linköping, Sweden

Tel.: +46 - (0)13 23 66 80
info@dynamore.se

www.dynamore.se
www.dynaexamples.com
www.dynasupport.com
www.dynalook.com

DYNAmore worldwide
Germany ▪ France ▪ Italy ▪ Sweden ▪ Switzerland ▪ USA

Find us on