AIT Lablink FMU Simulator

AIT Lablink Development Team

Apr 15, 2022

INSTALLATION

1 Installation **3**
- 1.1 Maven project dependency . . . 3
- 1.2 Installation from source . . . 3
- 1.3 Troubleshooting the installation . . . 4

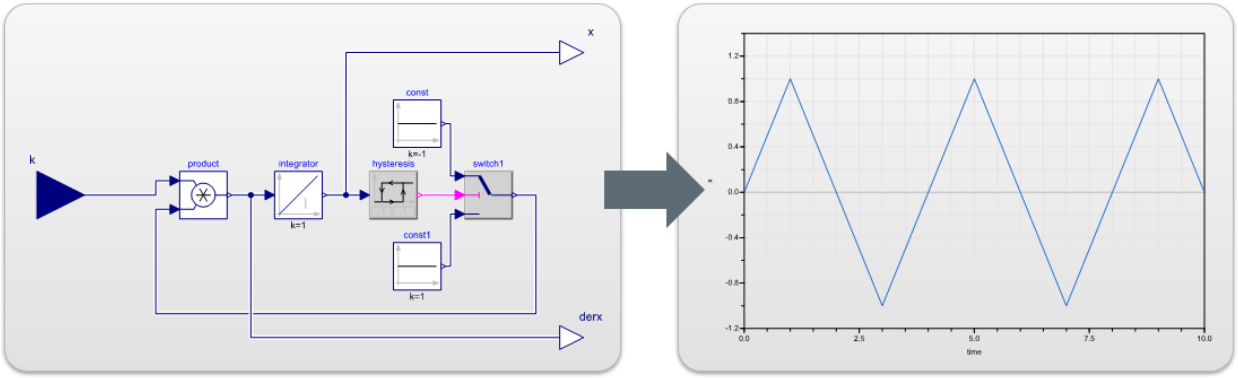
2 Running the clients **5**
- 2.1 Invoking the clients from the command line . . . 5

3 Configuration **7**
- 3.1 Overview . . . 7
- 3.2 Basic Lablink Client Configuration . . . 7
- 3.3 FMU Simulator Configuration . . . 8
- 3.4 Initial Value Configuration . . . 9
- 3.5 Input and Output Configuration . . . 9
- 3.6 Example Configuration . . . 9

4 Examples **11**
- 4.1 Prerequisites . . . 11
- 4.2 Example 1: Dynamic simulation using an FMU for Model Exchange . . . 12
- 4.3 Example 2: Fixed-step simulation using an FMU for Model Exchange . . . 13

This package provides Lablink clients that can import and execute Functional Mock-up Units (FMUs), i.e., co-simulation components compliant to the Functional Mock-up Interface (FMI). The clients in this package rely on functionality provided by the FMI++ Library.

Because FMUs may implement different features (model exchange and/or co-simulation) and because FMUs can be used in various synchronization schemes, this package provides several types of **FMU simulators**. Each FMU simulator has a different approach for synchronization (fixed step, discrete event, etc.) and its usefullness for any application depends highly on the specific requirements (especially the synchronization schedule).



The following FMU simulators are provided by this package:

- **DynamicFmuModelExchangeAsync**: This client simulates FMUs for Model Exchange and synchronizes them periodically to real time. At each synchronization point the client updates its output ports according to the selected FMU model ouput variables. In case an (internal) event is detected during the integration of the FMU model, the periodic synchronization schedule is adapted such that the output ports are updated at the corresponding event time. New inputs to the client are treated as (external) events and the integration of the FMU model and the synchronization schedule is adapted accordingly. *This FMU simulator is specifically suitable for discrete event simulations, where inputs are not periodical and/or internal events of the model are of special interest.*
- **FixedStepFmuModelExchangeAsync**: This client simulates FMUs for Model Exchange and synchronizes them strictly periodically to real time. Inputs to the model are delayed to the next synchronization point. In case an (internal) event is detected during the integration of the FMU model, the event is handled properly, but the outputs of the client are not updated. *This FMU simulator is specifically suitable for fixed-step simulations, where inputs and/or outputs of the model are periodical.*

CHAPTER

ONE

INSTALLATION

Find information about the installation of the Lablink FMU simulator clients *here*.

1.1 Maven project dependency

The Lablink FMU simulator client's compiled Java package is available on the Maven Central Repository. Use it in your local Maven setup by including the following dependency into your *pom.xml*:

```
<dependency>
  <groupId>at.ac.ait.lablink.clients</groupId>
  <artifactId>fmusim</artifactId>
  <version>0.0.1</version>
</dependency>
```

Note: You may have to adapt this snippet to use the latest version, please check the Maven Central Repository.

1.2 Installation from source

Installation from source requires a local **Java Development Kit** installation, for instance the Oracle Java SE Development Kit 13 or the OpenJDK.

Check out the project and compile it with Maven:

```
git clone https://github.com/ait-lablink/lablink-fmusim
cd lablink-fmusim
mvnw clean package
```

This will create JAR file *fmusim-<VERSION>-jar-with-dependencies.jar* in subdirectory *target/assembly*. Furthermore, all required FMI++ shared library files (DLLs, SOs) will be copied to subdirectory *target/natives*.

1.3 Troubleshooting the installation

Error message:

```
[ERROR] Failed to execute goal on project fmusim: Could not resolve dependencies for␣
↪project at.ac.ait.lablink.clients:fmusim:jar:0.0.1:
Could not find artifact at.ac.ait.fmipp:libfmipp:zip:natives-libfmipp-sundials-windows-
↪release-x64:0.0.1 in central (https://repo1.maven.org/maven2)
```

The FMU simulator requires the FMI++ Library to be installed, including the Java bindings. They should be avialable via the AIT Lablink repository. In case they are not installed (or not for your specific OS), you can install them yourself to your local Maven repository. To do so, follow the FMI++ instructions and additionally specify the CMake flag `JAVA_MAVEN_INSTALL:BOOL=ON` to automatically install the generated files to your local Maven repository. In this case, you will have to include your local Maven repository to your setup (see here).

CHAPTER

TWO

RUNNING THE CLIENTS

Find basic instructions for running the clients *here*.

2.1 Invoking the clients from the command line

When running the clients, the use of the `-c` command line flag followed by the URI to the configuration (see *here*) is mandatory. Furthermore, the path to the directory containing the FMI++ shared library files (by default in subdirectory *target/natives* when *building from source*) has to be added to the system path.

For example, on Windows this could look something like this:

```
SET FMIPP_DLL_DIR=\path\to\lablink-fmusim\target\natives
SET PATH=%FMIPP_DLL_DIR%;%PATH%

SET LLCONFIG=http://localhost:10101/get?id=
SET CONFIG_FILE_URI=%LLCONFIG%ait.test.fmusim.dynamic_me.fmu.config

SET FMUSIM=at.ac.ait.lablink.clients.fmu.DynamicFmuModelExchangeAsync
SET FMUSIM_JAR_FILE=\path\to\lablink-fmusim\target\assembly\fmusim-<VERSION>-jar-with-
↪dependencies.jar

java.exe -cp %FMUSIM_JAR_FILE% %FMUSIM% -c %CONFIG_FILE_URI%
```

CHAPTER

THREE

CONFIGURATION

Find the reference for writing a configuration for a Lablink FMU simulator client *here*.

3.1 Overview

The configuration has to be JSON-formatted. It is divided into the following categories:

Client basic configuration of the Lablink client

FMU basic configuration related to the FMU simulator

InitialValues configuration of initial values of the FMU instance

Input configuration of the client's inputs, each associated to an FMU input variable

Output configuration of the client's outputs, each associated to an FMU output variable

In the following, the configuration parameters for these categories are listed.

See also:

See *below* for an example of a complete JSON configuration.

3.2 Basic Lablink Client Configuration

Required parameters:

ClientName client name

GroupName group name

ScenarioName scenario name

labLinkPropertiesUrl URI to Lablink configuration

syncHostPropertiesUrl URI to sync host configuration

Optional parameters:

ClientDescription description of the client

ClientShell activate Lablink shell (default: `false`).

3.3 FMU Simulator Configuration

Required parameters:

URI URI to the FMU. Paths can be specified as `fmusim://relative/path/to/my.fmu`, where the path will be interpreted relative to the path specified via system property `dmuDir`. For instance, using option `-DfmuDir=/a/b/c` when starting the plotter client and specifiying `plotter://x/y/z.fmu` will result in CSV files being written to directory `/a/b/c/x/y/z.fmu`.

Optional parameters:

DefaulUpdatePeriod_ms default period of the synchronization schedule in milliseconds (default: `1000`)

LoggingOn turn on/off log messages from the FMU (default: `false`)

IntegratorType select integrator type (default: `bdf`):

`eu` Forward Euler method

`rk` 4th order Runge-Kutta method with constant step size

`abm` Adams-Bashforth-Moulton multistep method with adjustable order and constant step size

`ck` 5th order Runge-Kutta-Cash-Karp method with controlled step size

`dp` 5th order Runge-Kutta-Dormand-Prince method with controlled step size

`fe` 8th order Runge-Kutta-Fehlberg method with controlled step size

`bs` Bulirsch-Stoer method with controlled step size

`ro` 4th order Rosenbrock Method for stiff problems

`bdf` Backwards Differentiation formula from Sundials. This stepper has adaptive step size, error control and an internal algorithm for the event search loop. The order varies between 1 and 5. Well suited for stiff problems.

`abm2` Adams-Bashforth-Moulton method from sundials. This stepper has adaptive step size, error control, and an internal algorithm for the event search loop. The order varies between 1 and 12. Well suited for smooth problems.

ModelTimeScaleFactor simulation time scaling factor, i.e., speed-up or slow-down of progress of logical simulation time (default: `1`)

ModelStartTime_s start time (logical simulation time) for FMU model (default: `0`)

TimeDiffResolution_s resolution for resolving time differences in seconds (default: `1e-4`)

Optional parameters for **DynamicFmuModelExchangeAsync only** (for expert users):

NIntegratorSteps number of integration intervals within a synchronozation period (default: `2`)

NSteps number of integration steps within each integration interval for fixed-step integrators (default: `2`)

3.4 Initial Value Configuration

Configuration for each FMU model variable that should be initialized with a specific (non-default) value:

VariableName name of the FMU variable

DataType type of the FMU variable, allowed values are `double`, `long`, `boolean` and `string`

Value initial value

3.5 Input and Output Configuration

Required configuration parameters for each input/output:

VariableName name of the client's input/output port, has to correspond to an appropriate FMU variable

DataType data type of the client's input/output port, has to be compatible to the corresponding FMU variable type; allowed values are `double`, `long`, `boolean` and `string`

Optional configuration parameters for each input/output:

Unit unit associated to the client's input/output port

3.6 Example Configuration

The following is an example configuration for a *DynamicFmuModelExchangeAsync* client:

```
{
  "Client": {
    "ClientDescription": "FMU async simulator example.",
    "ClientName": "TestFMUAsync",
    "ClientShell": true,
    "GroupName": "FMUSimDemo",
    "ScenarioName": "FMUSimAsync",
    "labLinkPropertiesUrl": "http://localhost:10101/get?id=ait.all.all.llproperties",
    "syncHostPropertiesUrl": "http://localhost:10101/get?id=ait.test.fmusim.async.sync-
↪host.properties"
  },
  "FMU": {
    "DefaulUpdatePeriod_ms": 1000,
    "IntegratorType": "bdf",
    "TimeDiffResolution": 1e-06,
    "URI": "file:///C:/Development/lablink/lablink-fmusim/src/test/resources/zigzag.fmu"
  },
  "InitialValues": [
    {
      "DataType": "double",
      "Value": 0,
```

(continues on next page)

(continued from previous page)

```
      "VariableName": "integrator.y_start"
    },
    {
      "DataType": "double",
      "Value": 0.8,
      "VariableName": "k"
    }
  ],
  "Input": [
    {
      "DataType": "double",
      "Unit": "none",
      "VariableName": "k"
    }
  ],
  "Output": [
    {
      "DataType": "double",
      "Unit": "none",
      "VariableName": "x"
    },
    {
      "DataType": "double",
      "Unit": "none",
      "VariableName": "derx"
    }
  ]
}
```

CHAPTER

FOUR

EXAMPLES

Find step-by-step instructions for running the examples *here*.

4.1 Prerequisites

4.1.1 Required Lablink resources

The following Lablink resources are required for running the examples:

- Configuration Server: *config-0.0.1-jar-with-dependencies.jar*
- Datapoint Bridge: *dpbridge-0.0.1-jar-with-dependencies.jar*
- Lablink Plotter: *plotter-0.0.1-jar-with-dependencies*

When *building from source*, the corresponding JAR files will be copied to directory *target/dependency*.

Note: You also need a **local copy** of the FMU file zigzag.fmu. In case you have checked out a local copy of the lablink-fmusim repository, you can find it in the local subdirectory *src/test/resources*.

4.1.2 Starting the configuration server

Start the configuration server by executing script run_config.cmd in subdirectory examples/0_config. This will make the content of database file *test-config.db* available via http://localhost:10101.

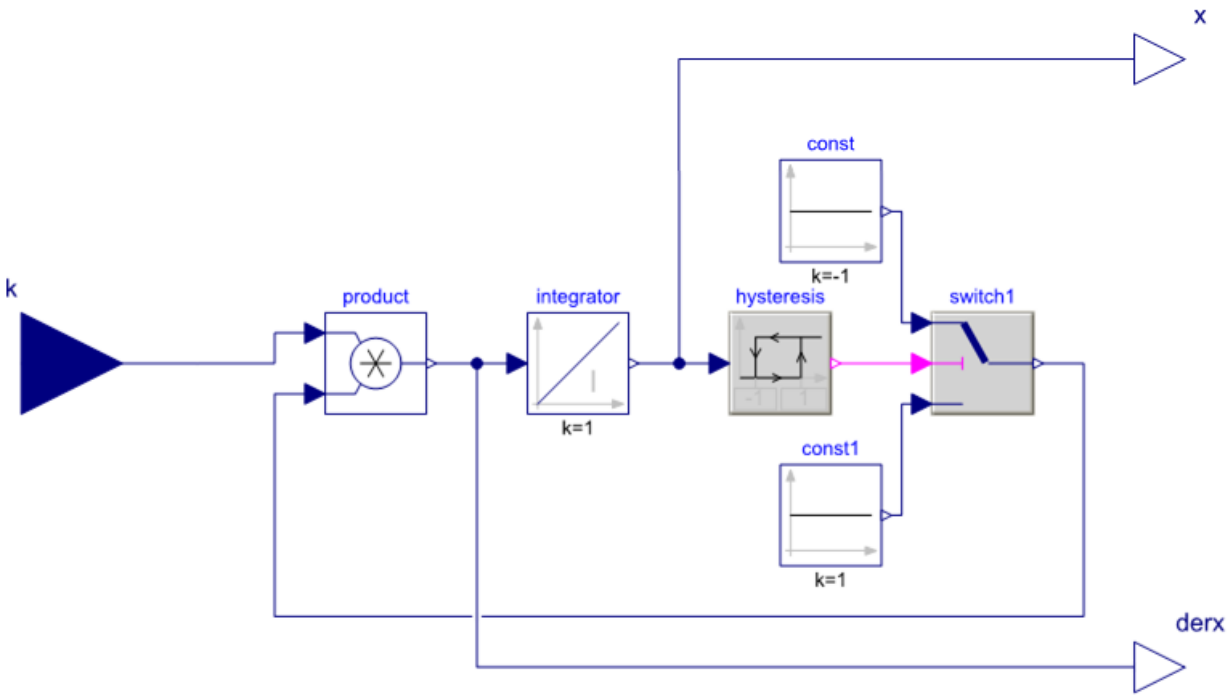
Note: Once the server is running, you can view the available configurations in a web browser via http://localhost:10101.

4.1.3 MQTT broker

An **MQTT broker** is required for running the example, for instance Eclipse Mosquitto or EMQ.

4.2 Example 1: Dynamic simulation using an FMU for Model Exchange

The figure below shows the graphical representation of the *zigzag* model used in this example. It comprises an integrator, whose input is a constant (model input variable `k`), and a hysteresis controller, which switches the sign of `k` in case the integrator's output variable `x` crosses a certain threshold (low: -1, high: 1). The resulting output is a zigzag pattern (hence the model name). The model has been compiled with Dymola into an FMU for Model Exchange, which can be found in subdirectory src/test/resources.



To run the example, edit attribute `FMU.URI` in the FMU simulator client configuration `ait.test.fmusim.dynamic_me.fmu.config` to match the absolut path of the FMU file (located in your local subdirectory *src/test/resources*). Use for instance DB Browser for SQLite to edit the configuration.

All relevant scripts can be found in subdirectory examples/1_dynamic_me. To run the example, execute all scripts either in separate command prompt windows or by double-clicking:

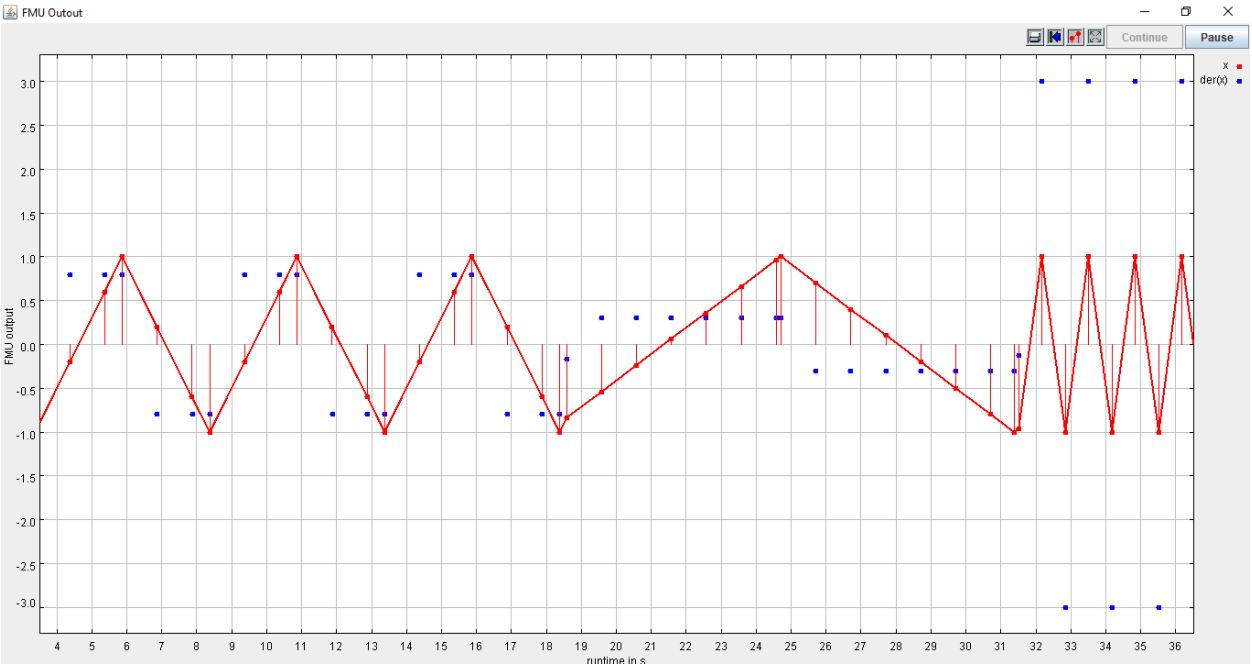
- fmusim.cmd: runs the FMU simulator
- dpb.cmd: runs the datapoint bridge service, connecting the FMU simulator and the plotter
- plot.cmd: runs the plotter, which will plot incoming data

Note: The order in which the scripts are started is arbitrary.

Once the FMU simulator client starts up, the client shell can be used to interact with the FMU model. For instance, input variable `k` of the FMU model can be changed:

```
llclient> svd k .3
Success
llclient> svd k 3
Success
```

An example of how the FMU reacts on these inputs can be seen in the following figure.



4.3 Example 2: Fixed-step simulation using an FMU for Model Exchange

This example uses the same *zigzag* model as the previous example.

To run the example, edit attribute `FMU.URI` in the FMU simulator client configuration `ait.test.fmusim.fixedstep_me.fmu.config` to match the absolut path of the FMU file (located in your local subdirectory *src/test/resources*). Use for instance DB Browser for SQLite to edit the configuration.

All relevant scripts can be found in subdirectory examples/2_fixedstep_me. To run the example, execute all scripts either in separate command prompt windows or by double-clicking:

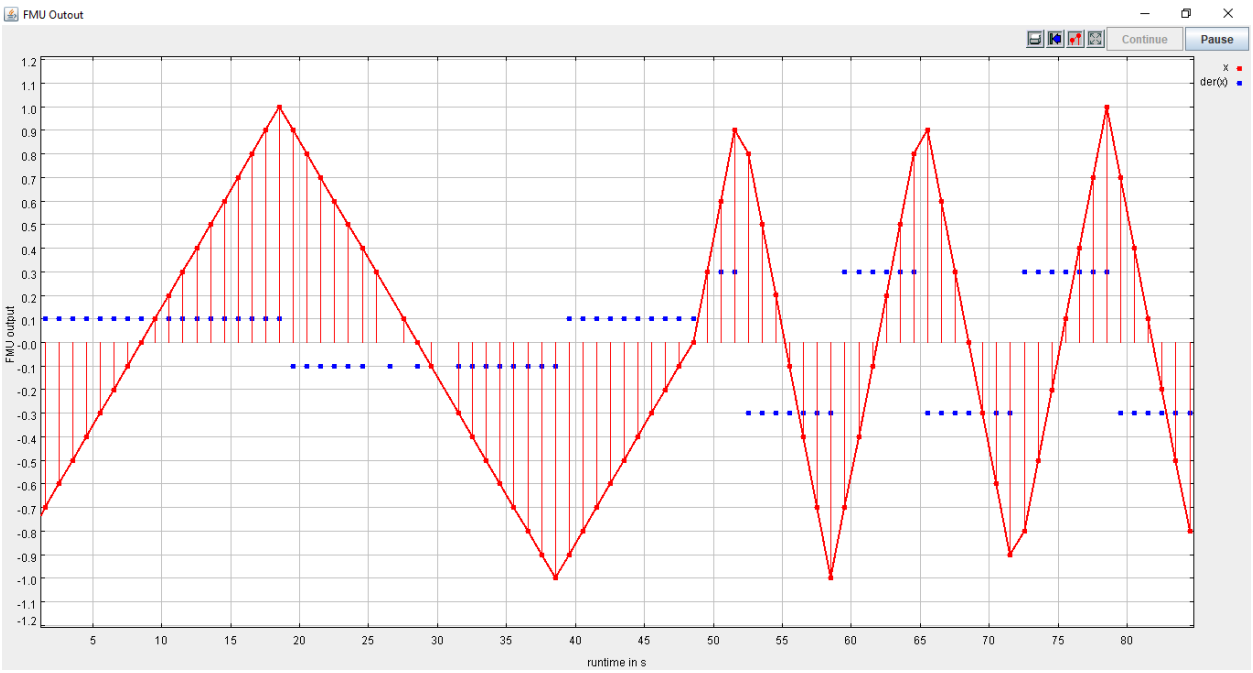
- fmusim.cmd: runs the FMU simulator
- dpb.cmd: runs the data point bridge service, connecting the FMU simulator and the plotter
- plot.cmd: runs the plotter, which will plot incoming data

Note: The order in which the scripts are started is arbitrary.

Once the FMU simulator client starts up, the client shell can be used to interact with the FMU model. For instance, input variable `k` of the FMU model can be changed:

```
llclient> svd k 0.3
Success
```

An example of how the FMU reacts on these inputs can be seen in the following figure. Notice the differences to the previous example, where the FMU simulator did update the outputs not only at strictly periodic intervals.


FMU Outout
Continue
Pause
x
der(x)
FMU output
runtime in s