

---

# **CyDER Master Algorithm**

***Release 1.0.0***

**LBL - Building Technology and Urban Systems Division**

Feb 13, 2017



## CONTENTS

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Installation and Configuration</b>	<b>3</b>
2.1	Running PyFMI with Python 3.4 on Windows 32 bit . . . . .	3
2.1.1	Requirements . . . . .	3
2.1.2	Compilation . . . . .	3
<b>3</b>	<b>Co-simulation</b>	<b>5</b>
<b>4</b>	<b>Notation</b>	<b>7</b>
<b>5</b>	<b>Glossary</b>	<b>9</b>
<b>6</b>	<b>Acknowledgments</b>	<b>11</b>
<b>7</b>	<b>Disclaimers</b>	<b>13</b>
<b>8</b>	<b>Copyright and License</b>	<b>15</b>
8.1	Copyright . . . . .	15
8.2	License Agreement . . . . .	15
	<b>Index</b>	<b>17</b>



## **INTRODUCTION**

This user manual explains how to install and use PyFMI to couple the CYMDISTToFMU with other FMUs such as a GridDyn FMU.



## INSTALLATION AND CONFIGURATION

This chapter describes how to install PyFMI on Windows.

### Running PyFMI with Python 3.4 on Windows 32 bit

*PyFMI* is a python package which can be used to import and run a CYMDIST FMU. In *PyFMI* version 2.3.1, a master algorithm was added to import and link multiple FMUs for co-simulation. At time of writing, there was no *PyFMI* 2.3.1 executable available for Python 3.4 for Windows 32bit (See [PyPyi](#)). The next steps describe requirements and steps to perform to compile *PyFMI* version 2.3.1 from source.

---

**Note:** To avoid having to recompile *PyFMI* dependent libraries from source, we recommend to use pre-compiled Windows binaries whenever available.

---

### Requirements

The next table shows the list of Python modules and softwares used to compile version 2.3.1 of PyFMI from source so it can run with Python 3.4 on Windows 32 bit.

Modules	Version	Link
Python 3.4	Windows 32 bit	<a href="https://www.python.org/ftp/python/3.4.3/python-3.4.3.msi">https://www.python.org/ftp/python/3.4.3/python-3.4.3.msi</a>
FMI Library	2.0.2	<a href="http://www.jmodelica.org/FMILibrary">http://www.jmodelica.org/FMILibrary</a>
Cython	0.25.1	<a href="https://pypi.python.org/pypi/Cython/0.25.1">https://pypi.python.org/pypi/Cython/0.25.1</a>
Numpy	1.11.2	<a href="https://pypi.python.org/pypi/numpy/1.11.2">https://pypi.python.org/pypi/numpy/1.11.2</a>
Scipy	0.16.1	<a href="https://sourceforge.net/projects/scipy/files/scipy/0.16.1">https://sourceforge.net/projects/scipy/files/scipy/0.16.1</a>
lxml	3.4.4	<a href="https://pypi.python.org/pypi/lxml/3.4.4">https://pypi.python.org/pypi/lxml/3.4.4</a>
Assimulo	2.7b1	<a href="https://pypi.python.org/pypi/Assimulo/2.7b1">https://pypi.python.org/pypi/Assimulo/2.7b1</a>
pyparsing	2.1.10	<a href="https://pypi.python.org/pypi/pyparsing/2.1.10">https://pypi.python.org/pypi/pyparsing/2.1.10</a>
matplotlib	1.4.3	<a href="https://pypi.python.org/pypi/matplotlib/1.4.3">https://pypi.python.org/pypi/matplotlib/1.4.3</a>
C-Compiler	Microsoft Visual Studio 2010 Pro	
PyFMI	2.3.1 (source)	<a href="https://pypi.python.org/pypi/PyFMI">https://pypi.python.org/pypi/PyFMI</a>

### Compilation

To compile *PyFMI* from source, run

```
python setup.py install -fmil-home=path_to_FMI_Library\
```

where `path_to_FMI_Library\` is the path to the FMI library.





## CO-SIMULATION

This section explains how to link a CYMDIST FMU with another FMU for co-simulation. In this section, we used the GridDyn FMU for the simulation coupling.

The following code snippet shows how to import and link a CYMDIST FMU (CYMDIST.fmu) with a GridDyn FMU (GridDyn.fmu).

Line 1 and 2 import the *PyFMI* modules which are needed for the coupling.

Line 8 loads the CYMDIST FMU

Line 9 loads the GridDyn FMU

Line 11 defines a vector with the CYMDIST and the GridDyn FMUs models.

Line 12 defines the connections between the CYMDIST and the GridDyn FMUs (gridyn, "VMAG\_A", cymdist, "VMAG\_A") means that the output VMAG\_A of the GridDyn FMU is connected to the input VMAG\_A of the CYMDIST FMU.

Line 25 passes the FMUs models and their connection to the master algorithm.

Line 27 gets the simulation option object.

Line 28 sets the communication step size.

Line 29 sets the logging to true.

Line 31 invokes the function which is used to simulate the coupled models.

```
1  from pyfmi import load_fmu
2  from pyfmi.master import Master
3
4  def simulate_multiple_fmus():
5      """Simulate one CYMDIST FMU coupled to a GridDyn FMU.
6
7      """
8      cymdist=load_fmu("CYMDIST.fmu", log_level=7)
9      gridyn=load_fmu("GridDyn.fmu", log_level=7)
10
11     models = [cymdist, gridyn]
12     connections = [(gridyn, "VMAG_A", cymdist, "VMAG_A"),
13                   (gridyn, "VMAG_B", cymdist, "VMAG_B"),
14                   (gridyn, "VMAG_C", cymdist, "VMAG_C"),
15                   (gridyn, "VANG_A", cymdist, "VANG_A"),
16                   (gridyn, "VANG_B", cymdist, "VANG_B"),
17                   (gridyn, "VANG_C", cymdist, "VANG_C"),
18                   (cymdist, "KWA_800032440", gridyn, "KWA_800032440"),
19                   (cymdist, "KWB_800032440", gridyn, "KWB_800032440"),
20                   (cymdist, "KWC_800032440", gridyn, "KWC_800032440"),
```

```
21         (cymdist, "KVARA_800032440", gridyn, "KVARA_800032440"),
22         (cymdist, "KVARB_800032440", gridyn, "KVARB_800032440"),
23         (cymdist, "KVARC_800032440", gridyn, "KVARC_800032440"),]
24
25     coupled_simulation = Master (models, connections)
26
27     opts=coupled_simulation.simulate_options()
28     opts['step_size']=1.0
29     opts['logging']=True
30
31     res=coupled_simulation.simulate(options=opts,
32                                   start_time=0.0,
33                                   final_time=1.0)
34
35 if __name__ == '__main__':
36     simulate_multiple_fmusc()
```

## NOTATION

This chapter shows the formatting conventions used throughout the User Guide.

The command-line is an interactive session for issuing commands to the operating system. Examples include a DOS prompt on Windows, a command shell on Linux, and a Terminal window on MacOS.

The User Guide represents a command window like this:

```
# This is a comment.  
> (This is the command prompt, where you enter a command)  
(If shown, this is sample output in response to the command)
```

Note that your system may use a different symbol than “>” as the command prompt (for example, “\$”). Furthermore, the prompt may include information such as the name of your system, or the name of the current subdirectory.



## GLOSSARY

**Dymola** Dymola, Dynamic Modeling Laboratory, is a modeling and simulation environment for the Modelica language.

**Functional Mock-up Interface** The Functional Mock-up Interface (FMI) is the result of the Information Technology for European Advancement (ITEA2) project *MODELISAR*. The FMI standard is a tool independent standard to support both model exchange and co-simulation of dynamic models using a combination of XML-files, C-header files, C-code or binaries.

**Functional Mock-up Unit** A simulation model or program which implements the FMI standard is called Functional Mock-up Unit (FMU). An FMU comes along with a small set of C-functions (FMI functions) whose input and return arguments are defined by the FMI standard. These C-functions can be provided in source and/or binary form. The FMI functions are called by a simulator to create one or more instances of the FMU. The functions are also used to run the FMUs, typically together with other models. An FMU may either require the importing tool to perform numerical integration (model-exchange) or be self-integrating (co-simulation). An FMU is distributed in the form of a zip-file that contains shared libraries, which contain the implementation of the FMI functions and/or source code of the FMI functions, an XML-file, also called the model description file, which contains the variable definitions as well as meta-information of the model, additional files such as tables, images or documentation that might be relevant for the model.

**Modelica** Modelica is a non-proprietary, object-oriented, equation-based language to conveniently model complex physical systems containing, e.g., mechanical, electrical, electronic, hydraulic, thermal, control, electric power or process-oriented subcomponents.

**MODELISAR** MODELISAR is an ITEA 2 (Information Technology for European Advancement) European project aiming to improve the design of systems and of embedded software in vehicles.

**PyFMI** PyFMI is a package for loading and interacting with Functional Mock-Up Units (FMUs), which are compiled dynamic models compliant with the Functional Mock-Up Interface (FMI).

**Python** Python is a dynamic programming language that is used in a wide variety of application domains.



## ACKNOWLEDGMENTS

The development of this documentation was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technologies of the U.S. Department of Energy, under contract No. xxx.

The following people contributed to the development of this program:

- Thierry Stephane Noudui, Lawrence Berkeley National Laboratory
- Jonathan Coignard, Lawrence Berkeley National Laboratory
- Michael Wetter, Lawrence Berkeley National Laboratory





## **DISCLAIMERS**

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor The Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or The Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or The Regents of the University of California.



## COPYRIGHT AND LICENSE

### Copyright

“CyDER v01” Copyright (c) 2017, The Regents of the University of California, through Lawrence Berkeley National Laboratory (subject to receipt of any required approvals from the U.S. Dept. of Energy). All rights reserved.

If you have questions about your rights to use or distribute this software, please contact Berkeley Lab’s Innovation & Partnerships Office at [IPO@lbl.gov](mailto:IPO@lbl.gov).

NOTICE. This Software was developed under funding from the U.S. Department of Energy and the U.S. Government consequently retains certain rights. As such, the U.S. Government has been granted for itself and others acting on its behalf a paid-up, nonexclusive, irrevocable, worldwide license in the Software to reproduce, distribute copies to the public, prepare derivative works, and perform publicly and display publicly, and to permit others to do so.

### License Agreement

“CyDER v01” Copyright (c) 2017, The Regents of the University of California, through Lawrence Berkeley National Laboratory (subject to receipt of any required approvals from the U.S. Dept. of Energy). All rights reserved.

Redistribution and use in source and binary forms, with or without modification,

are permitted provided that the following conditions are met:

- (1) Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- (2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- (3) Neither the name of the University of California, Lawrence Berkeley National Laboratory, U.S. Dept. of Energy nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

You are under no obligation whatsoever to provide any bug fixes, patches, or upgrades to the features, functionality or performance of the source code (“Enhancements”) to anyone; however, if you choose to make your Enhancements available either publicly, or directly to Lawrence Berkeley National Laboratory, without imposing a separate written license agreement for such Enhancements, then you hereby grant the following license: a non-exclusive, royalty-free, perpetual license to install, use, modify, prepare derivative works, incorporate into other computer software, distribute, and sublicense such enhancements or derivative works thereof, in binary and source code form.

## D

Dymola, [9](#)

## F

Functional Mock-up Interface, [9](#)

Functional Mock-up Unit, [9](#)

## M

Modelica, [9](#)

MODELISAR, [9](#)

## P

PyFMI, [9](#)

Python, [9](#)