CyDER - An FMI-based Co-Simulation Platform for Distributed Energy Resources

Thierry S. Nouidui[a], Jonathan Coignard[a], Christoph Gehbauer[a], Michael Wetter[a], Jhi-Young Joo[b], Evangelos Vrettos[a]

[a]Lawrence Berkeley National Laboratory, Berkeley, CA, USA; [b] Lawrence Livermore National Laboratory, Livermore, CA, USA

ARTICLE HISTORY
Compiled October 8, 2018

ABSTRACT
The increased integration of Distributed Energy Resources (DERs) is bringing a number of challenges to the power grid. These include reverse power flows in distribution systems and potentially transmission systems, and grid stability. So far, specialized tools have been developed to capture some of the impact of DERs at the distribution level. However, Distribution System Operators (DSOs) lack visibility into the overall system conditions. Furthermore, the impact of increasing DERs is not limited to the distribution level but also influences the transmission grid. To support the planning and operation of the grid, we developed a co-simulation platform called CyDER (A Cyber Physical Co-simulation Platform for Distributed Energy Resources in Smart Grids) that integrates various domain-specific simulation tools. CyDER is based on the Functional Mock-up Interface (FMI) standard. This paper gives an overview of CyDER and demonstrates its use based on two applications.

KEYWORDS
Functional Mock-up Interface; Functional Mock-up Unit; Co-simulation; Power Grid; Distributed Energy Resources; Modelica.

1. Introduction

The complexity of distribution grids is drastically increasing as a result of integrating larger shares of distributed generation and storage devices, uncertainties in renewable power generation, and advanced electronics-based controllers (Hadjsaid and Sabonnadiere 2012). Solar photovoltaic (PV) systems are currently the renewable energy technology with the highest growth rate in many countries worldwide. Especially in many states in the U.S., federal tax credit and reduced in solar panel manufacturing costs have boosted adoption of solar PV systems by end-users. Notably, the total installed PV power at end-users premises in distribution grids exceeds 4,000 MW (Solar Energy Industries Association 2017). Since the peak PV generation does not necessarily coincide with the peak load demand, the power flow patterns are changing and some distribution feeders are facing voltage stability issues. Besides PV, increasing adoption of Electric Vehicles (EVs) and their uncontrolled charging can also overload existing distribution system equipment, by amplifying the peak load demand in the

Corresponding Author: Thierry S. Nouidui: Email: thierry.nouidui@yahoo.de

evening hours when PV generation decreases and users start plugging in their vehicles.

The widespread adoption of PV systems and EVs creates new concerns and problems for many Distribution System Operators (DSOs), as they need to reevaluate their planning strategies. In general, DSOs analyze potential impacts on the grid when and where they anticipate changes in their systems, or on an event basis, for example, in case of higher peak loads, equipment failures, or increased requests for PV installation approvals. Indicative of the general concerns is the fact that the California Energy Commission has ordered the States investor-owned utilities to conduct integration capacity analysis on their feeders to assess maximum hosting capacities of additional load and distributed generation (CPUC 2016).

The challenges in distribution system planning with high penetration of Distributed Energy Resources (DERs) are multi-fold. First, there are various DER technologies being integrated in the system such as PVs, EVs, storage devices, and demand response. The integration levels of each technology can widely vary depending on the geographic location, whereas their impacts heavily depend on the feeder topology. Secondly, DER integration increases the coupling between the distribution grids and the transmission system. On sunny days with low load demand, some feeders with PV generation are already experiencing reverse power flows towards the transmission grid. The problem here is that the software tools typically used by DSOs for distribution grid planning are different to those used by Transmission System Operators (TSOs) for transmission grid planning, which further complicates a combined transmission- and distribution-level system analysis.[1] Thirdly, with increased penetration of DERs, the operational uncertainty in the grid also increases due to variable weather patterns and user behavior. Last, large-scale DER integration increases the interactions and dependencies between the power system and other infrustructure systems, such as communication systems, markets, information systems for data analytics, heating/cooling systems, buildings, etc.

The aforementioned challenges necessitate a modular, extensible, and scalable power system planning tool. Such a tool should enable the coupling of different software modules that model various components of distribution system layer. The tool should also be able to handle uncertainties in PV generation and EV charging. Moreover, interactions and data exchange among the various systems and components need to be represented in the tool by modeling the associated communication networks. Overall, DSOs need a new simulation platform that supports operational tasks, increases situational awareness, and facilitates decision making on new investments in equipment or control technologies.

In principle, there exist two general approaches for the development of such a multi-domain and multi-purpose simulation tool. The first approach would be to extend existing simulation tools to include all additional desired functionalities that are currently not supported. However, this approach is not practical and is limited by the vision of the tool's developer, anticipated market placement, and availability of personnel and financial resources. The second approach, which we see as promising, is co-simulation of independent simulators on a common platform. The advantage of this approach is that specialized software and third-party tools can be leveraged to study complex interdependencies between systems while preserving simplicity, transparency, flexibility, and scalability of the simulation environment (Chatzivasileiadis et al 2015).

Although co-simulation tools are relatively new to power systems applications, sev-

[1]The term transmission system refers to high-voltage networks for long-distance power transmission (nominal voltage is typically larger than 115 kV), whereas the term distribution system refers to medium- or low-voltage networks for local power distribution (nominal voltage is typically smaller than 69 kV).

eral co-simulation platforms are presented in the literature (Gomes et al. 2017). However, most of them involve only two simulation tools, and are set-up ad-hoc without a clear semantics of how tools are synchronized, which is far from a scalable multi-domain platform for cyber-physical systems such as power systems.

Recent development led to few co-simulation power systems platforms which support multiple simulation tools. These include the Framework for Network Co-simulation (FNCS) (Daily et al. 2014), which wraps simulation engines as agents that communicate through a FNCS broker; the platform SCEPTRE which wraps simulation tools and control devices in virtual machines that communicate through network (SCEPTRE 2016); the Integrated Energy System Modeling Framework (IEMS) which uses the Discrete Event System Specification (DEVS) (Zeigler et al 1993) formalism to wrap different simulation tools which then communicate through a DEVS coordinator (Mittal et al 2015); the Integrated Grid Modeling System (IGMS) which is based on ZeroMQ (ZeroMP 2015) and defines peer-to-peer interfaces for multi-domain co-simulation (Palmintier et al 2017); the Mosaik framework which defines an Application Programming Interface (API) to be implemented by a simulation tool so it can be executed as an individual process that is coordinated by the Mosaik coordinator (Buesher et al 2014); and the Hierarchical Engine for Large-scale Infrastructure Co-Simulation (HELICS) which defines an API to be implemented by a simulation tool so it can be co-simulated with other tools within HELICS. Other co-simulation efforts in the area of electric power systems include the work done by Roche et al (2012), Godfrey et al (2010), or Gomez-Gualdron et al (2006).

Although there is a momentum and need in developing co-simulation platforms for power systems, one key limitation of the current trend is that the development of existing platforms is not built on open standards. They usually require simulation tools to implement their specific API, which in turn requires ad-hoc solutions for interoperability. Furthermore, they are prone to non-determinism because of lack of rigor in the semantics, and can therefore quickly become hard to maintain or obsolete because of lack of industry support and trust.

The goal of the CyDER platform is to use a well defined open industry standard to couple power system tools in different time domains and voltage levels, such as quasi-static distribution and transmission system simulations, and real-time digital simulations, with tools that are not developed within the traditional scope of power systems. These tools can simulate various domains such as whole buildings energy use, PV generation, and EVs on a high level, down to more detailed models of Heating, Ventilation and Air-Conditioning (HVAC) systems, PV modules, or Electronic Control Units (ECUs) of EVs. The backbone, which ultimately enables the proposed co-simulation of the various tools, is the Functional Mock-up Interface (FMI) standard (Blockwitz et al. 2012), an open industry standard which is well positioned to become the de facto standard for co-simulation.

Our contribution is as follows:

- A scalable co-simulation platform which uses an open industry standard to seamlessly integrate transmission and distribution system simulation tools for planning, analysis and operation of the grid.
- A collection of open-source transmission, distribution, and DERs tools which are compliant with the FMI interface.
- A software utility that allows to export new power systems simulation tools as input/output blocks, which are compliant with the FMI standard.

Note that The contribution of CyDER is not in the API (FMI standard), but in its application for power systems co-simulations and the development of Python modules to facilitate connecting simulators and launching co-simulations with the PyFMI master algorithm.

The rest of the paper is organized as follows. Section 2 discusses the CyDER technology. It describes a suite of tools developed in CyDER and the master algorithm used for the co-simulation of the CyDER modules. Section 3 presents two application examples where CyDER is used for power system planning analysis. The control algorithms used in these examples are simple and the associated results are merely indicative of CyDER's capabilities as a co-simulation platform. Section 4 concludes the paper with an outlook for future work.

2. Technical approach

CyDER (Fig. 1) is a co-simulation platform which integrates transmission and distribution system simulations, real-time data collection, power generation and load forecasting, EV charging forecasting, and real-time control of solar PV to predict, analyze, and accommodate high levels of PV penetration. Applications of CyDER include support of Hardware-In-the-Loop (HIL) simulation for controls development, contingency analysis, long-term infrastructure and policy planning analysis.

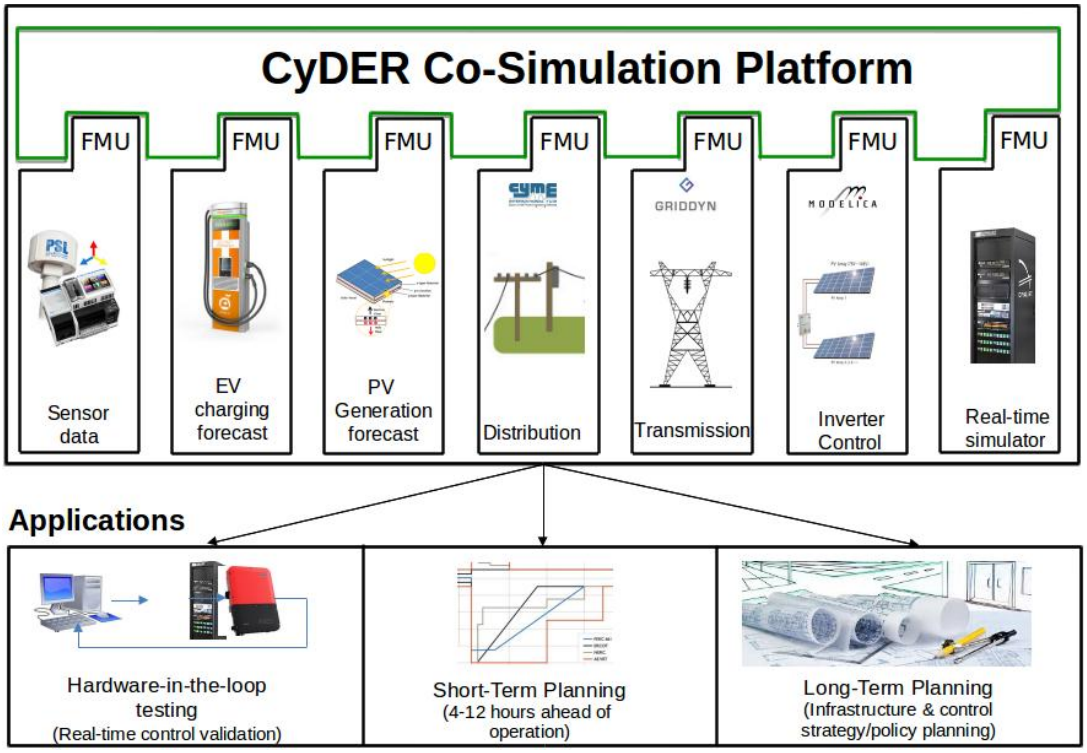

Figure 1. CyDER co-simulation platform with targeted applications.

CyDER leverages the FMI standard to seamlessly integrate different software modules in a using a standardized Application Programming Interface(API) with a well defined semantics. In this section, we first introduce the FMI-Standard. Second, we describe the software modules we developed in CyDER to support transmission and

distribution grid co-simulation. Third, we describe the master algorithm which is used to integrate the CyDER's software modules.

2.1. *FMI standard*

The FMI standard is the result of the ITEA2 MODELISAR project (Modelisar 2008), a 27-million-euro European project driven by the automotive industry. The objective of the project was to facilitate the exchange and interoperability of simulation models from different Original Equipment Manufacturers (OEMs) so they can be easily integrated to simulate a whole automobile prior to production. Since OEM component models are written in various languages, it was costly and time-consuming to integrate component models from different sources. To address this issue, the open standard FMI was created.

The FMI standardizes an API to be implemented by a simulation model described by differential, algebraic, and discrete equations with the goal of facilitating its interoperability with other models. The first version of FMI (1.0) was released in 2010 with support of more than 30 tools. The second version (2.0) was released in 2014 with support of more than 100 tools. A simulation model that implements the FMI standard is called a Functional Mock-up Unit (FMU). An FMU is an input/output representation of the model which is distributed as a zip file with the extension *.fmu*. This zip file contains an XML model description file with meta-data about the model, the model itself, and a set of C functions with standardized signatures and semantics to interact with the model. These functions are typically available as compiled C code, which may or may not include a solver. Optionally, source code can be provided. Modeling tools that support FMI typically contain an export functionality that exports a model as an FMU.

Fig. 2 shows a model described by a first order differential equation which has been exported as an FMU for model exchange (top), and an FMU for co-simulation (bottom). The model has one continuous state x, an input u, and an output y. When the model is exported as an FMU for model exchange, the FMU exposes at the FMI interface as inputs the time t, state x, and input u of the model. As outputs, the model exposes the state derivative and the output y. The FMU does not provide an integrator to integrate the state derivative. Therefore, an integrator needs to be provided by the tool which imports the FMU to compute the state trajectory outside of the FMU. When the model is exported for co-simulation, the FMU exposes at the interface as inputs of the model the time t, the suggested communication step size Δt and the input u. As outputs, it exposes, the integrated state x and the output y. In this situation, the FMU contains an integrator that computes the state trajectory from time t to time $t + \Delta t$, with Δt being proposed by the tool that imports and runs the FMU.

Note that the FMI standard does not prescribe how to coordinate multiple FMUs. This is left to a master algorithm as the efficient coordination is problem or application domain dependent.

2.2. *CyDER Software Modules*

The next section discusses a suite of modules and tools developed in CyDER to support model-based analysis of the power grid. All software modules are exported as FMUs to ease their interoperability. Unless specified, all software modules developed in CyDER

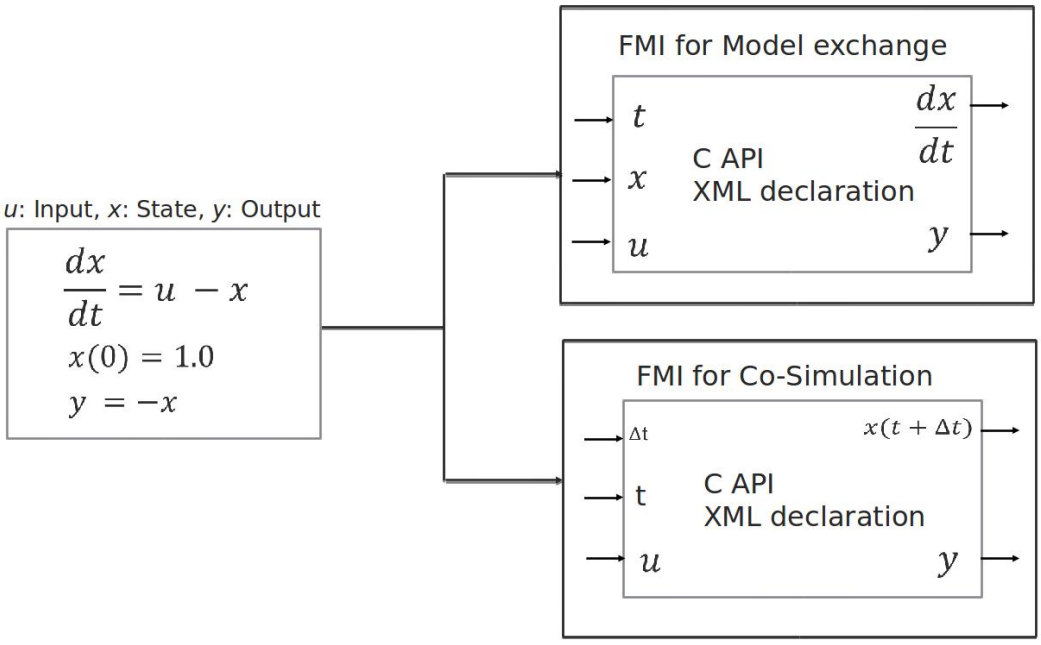

Figure 2. Example of a first-order model exported as an FMU for model exchange and co-simulation.

support FMI versions 1.0 and 2.0, and FMI APIs model exchange and co-simulation.

2.2.1. SimulatorToFMU

Although not widespread, more and more simulation tools such as CYMDIST[2] or PowerFactory[3] provide high-level APIs, which allow the tool to be controlled by external tools. These APIs are often written in Python.

To streamline the process of exporting such simulation tools as FMUs, SimulatorToFMU was developed. SimulatorToFMU is a Python utility which allows to export Python functions as FMUs. The approach of SimulatorToFMU is to wrap the high-level Python API in a Python function, which can then be exported as FMUs.

SimulatorToFMU uses a Python templating engine called jinja2[4] to generate a Modelica[5] block that contains a wrapper which communicates with the simulation tool through its Python API. It then invokes a Modelica translator which compiles the model to C-code and exports it as an FMU. Leveraging Modelica compilers to export Python functions as FMUs ensures the backward and forward compatibility of SimulatorToFMU with older and newer versions of the FMI specification as they become available in Modelica tools. Fig. 3 shows the high-level workflow for exporting a Simulator as an FMU.

SimulatorToFMU gets as input the path to an XML file which specifies the input and the output of the simulator as well as the Python function which interacts with the simulator. It then generates a Simulator FMU which is compliant with the FMI standard. Details about SimulatorToFMU can be found in Nouidui and Wetter (2018). SimulatorToFMU is open-source and freely available at https://github.com/LBNL-

[2]https://www.cyme.com/
[3]https://www.digsilent.de/en/powerfactory.html
[4]http://jinja.pocoo.org/
[5]https://www.modelica.org/

ETA/SimulatorToFMU.

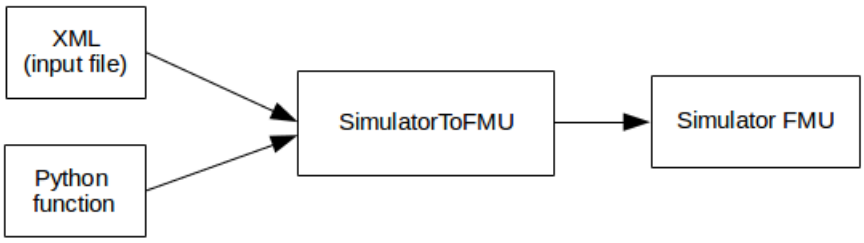

Figure 3. Exporting a Simulator as an FMU using SimulatorToFMU.

2.2.2. Distribution Grid FMU

CYMDIST was selected as the distribution planning tool of CyDER. CYMDIST was chosen because of its wide use by US utilities. CYMDIST provides a Python API which allows it to be exported as an FMU. To export CYMDIST as an FMU, a Python function which interfaces with the Python API of CYMDIST was developed. This function allows us to start CYMDIST, and to set values to, and get values from a running CYMDIST model at discrete times. SimulatorToFMU was used to export the Python function as an FMU.

2.2.3. Transmission Grid FMU

We selected the open source simulator GridDyn (Top 2016) as the transmission network simulator. We developed a utility to export a transmission network model as an FMU for co-simulation 2.0. The export utility is a small executable that gets as input an XML input file (which specifies the inputs and outputs that should be exposed through the FMI interface) along with the network model, and exports the model as an FMU.

2.2.4. Inverter Control FMU

To support use cases in which a PV inverter injects reactive power, a Volt-Var control was exported as an FMU. The controller was developed in Modelica. Input is the voltage v, which regulates the reactive power of the inverter. The control can be freely configured to different set-points. The model shown in Fig. 4 is an example that was configured to saturate at voltages below 0.95 Vp.u. and above 1.05 Vp.u, with a dead-band introduced from 0.99 Vp.u. to 1.01 Vp.u. The resulting control output starts at 100% capacitive as the system voltage is below the lower saturation threshold. As the system voltage increases, it reaches the intersection point where the reactive output starts to ramp down until it reaches the dead-band. By further increasing the voltage, the control starts to exceed the dead-band and ramp up on the inductive area until it saturates at the maximal voltage. The Modelica model was exported as an FMU using Dymola 2018 FD01[6], a commercial Modelica simulator.

[6]https://www.3ds.com/products-services/catia/products/dymola/

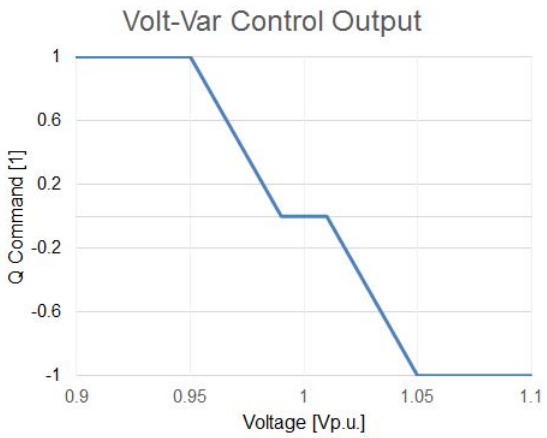

Figure 4. Volt-Var inverter control.

2.2.5. Real-Time Simulator FMU

To support Hardware-in-the-Loop (HIL) applications, we exported the real-time simulator OPAL-RT as an FMU. OPAL-RT is a simulation engine which is largely used for HIL applications. It provides a Python API which allows to compile, load, execute a model, and exchange data with the model for real-time simulation. An FMI interface was developed around the Python API and then exported by SimulatorToFMU.

To validate the work flow, a CYMDIST model was converted into an OPAL-RT model and tested in real-time. A master algorithm was used to vary control signals of loads in the system in a real-time simulation and the resulting changes in nodal voltages were read back. Fig. 5 shows results of the validation case where the FMU was run for 4 seconds. At every second, the control signal was switched from on to off. Fig. 5 shows how the voltage at a node changes as a function of the control signal.

2.2.6. Inverter API FMU

To communicate with physical hardware, an inverter API FMU was developed. For this purpose, a Python-based Hypertext Transfer Protocol (HTTP) minimal web server was utilized. This web server is a gateway which interfaces between the inverters, communicating via MODBUS, and external communication, via HTTP. The advantage of such HTTP-based commands is that host and client computers can be located anywhere. To interface with the web-server, a Python script was developed which allows to send data to and receive data from physical devices. SimulatorToFMU was used to export the Python script as an FMU which streamlines the process of interfacing with physical inverters through the FMI interface.

2.2.7. μPMU FMU

To communicate and retrieve sensor data from micro-Synchrophasor Measurement Units (μPMUs), we developed an HTTP web server similar to the one for the Inverter API FMU. The web server is an interface to the database which contains data measured by the μPMU. A python script to interface with the web server was developed and exported as an FMU using SimulatorToFMU. This standardizes the process of querying data from the database, and allows flexible deployment across different platforms.

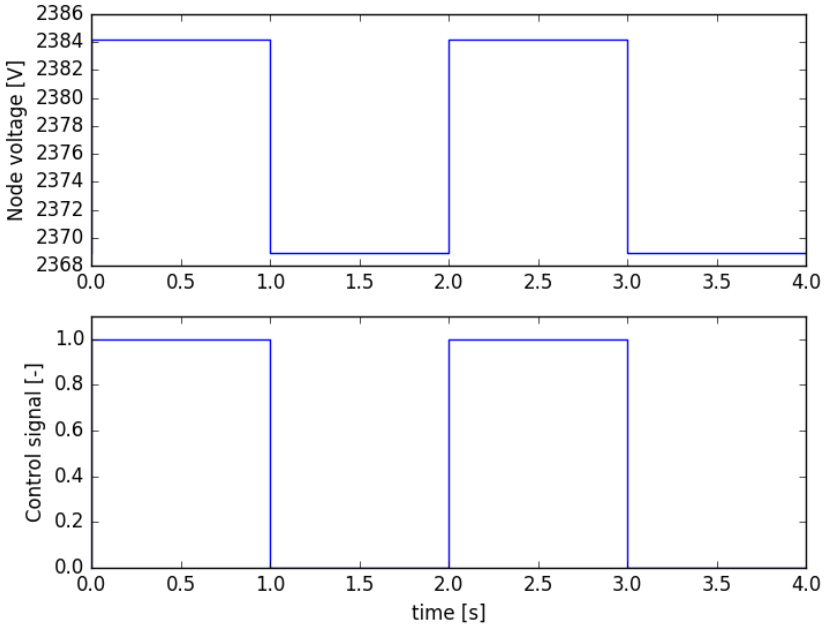

Figure 5. Validation of the export of an OPAL-RT model as an FMU.

2.2.8. EV FMU

In order to model Electric Vehicles (EVs) connected to the grid, we have created an FMU that simulates the power demand from a fleet of EVs based on individual vehicle information such as

(1) parking duration (hours)
(2) maximum charging power (kW)
(3) energy requirement (kWh)

The FMU also provides the opportunity to apply a volt-watt control to reduce voltage issues on the grid by shifting power demand from EVs while preserving energy constraints set by EV owners. Figure 6 shows different paths to charge an EV during the time it is parked. The green curve is for the case where the vehicle charges as soon as possible. The blue curve is for the case where the vehicle charges as late as possible, and the orange curve is for the case where the vehicle charges with a constant power demand. The EV load demand module was exported as an FMU using SimulatorToFMU.

2.2.9. PV FMU

To model PV generation, an empirical PV model was developed in the modeling language Modelica and exported by a Modelica simulator as an FMU. The PV generation P_{PV} is computed as

$$P_{PV} = A \cdot eta \cdot G \cdot (1 - S) \ , \tag{1}$$

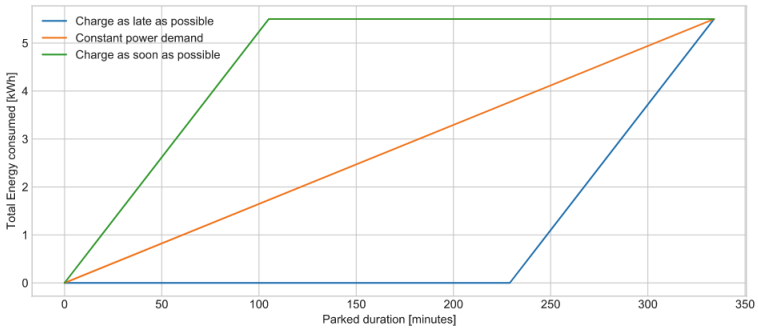

Figure 6. Flexibility in electric vehicle charging profile

where A is the area of the PV in m^2, eta is the lumped PV conversion efficiency including module and inverter, G is the incident solar irradiation at the tilted surface, and S is the percentage of shaded area. The output of the PV FMU is the PV power generation at the point of grid interconnection, in W, for a specific time of the day. To compute the incident solar irradiation, the model uses historical weather data in form of a Typical Meteorological Year (TMY) weather data file. The weather data is fed to a detailed sky model along with tilt, azimuth, latitude, longitude, and time zone to compute the incident solar irradiation on the PV surface. The model exposes a string parameter which specifies the path to the weather file. Parameters of the FMU that can be changed prior to simulation are the path to weather file, tilt, azimuth, and area of the PV.

2.3. *Master Algorithm*

To connect and simulate FMUs developed in CyDER, a master algorithms is needed. We selected PyFMI as the master algorithm for linking, exchanging, and synchronizing data between the FMUs at runtime. PyFMI is an open-source master algorithm written in Python. It provides functions which allow to interface with FMUs. The main functions are unzipping, loading, instantiating, initializing, and running the FMUs, while synchronizing the data exchange between the FMUs. Below is a pseudo code snippet which shows how PyFMI is used to load, connect, and run an inverter control FMU with the distribution grid simulator CYMDIST. Both FMUs implement the model exchange API of FMI 2.0. The steps involved in the process are:

- Load the FMUs (Line 9 and 10)
- Create an object which contains the FMUs instances (Line 12)
- Define the connection between the FMUs (Line 15)
- Create an object which represents the coupled system (Line 18)
- Simulate the coupled system (Line 21)
- Retrieve the simulation results (Line 24 and 25)

```
1 from pyfmi import load_fmu
2 from pyfmi.fmi_coupled import CoupledFMUModelME2
3
4 # Simulation parameters
5 start_time = 0.0
6 stop_time  = 0.1
7
8 # Loading the FMUs
```

```
cymdist=load_fmu("cymdist.fmu")
control=load_fmu("control.fmu")

models = [("cymdist", cymdist), ("control", control)]

# Creating connection list
connections = [(cymdist, "v", control, "v"), (control, "Q", cymdist, "Q")]

# Creating the coupled system
coupled_simulation = CoupledFMUModelME2 (models, connections)

# Running the simulation
res=coupled_simulation.simulate(start_time=start_time, final_time=stop_time)

# Retrieving the trajectories
sim_time=res["time"]
cymdist_v=res["cymdist.v"]
```

For the initialization of the coupled system, the master algorithm supports initialization based on graph cycle detection (Andersson 2016), which detects cycles by analyzing the dependency information between inputs and outputs of the coupled system. These input-output dependencies can be optionally provided by the FMUs. The graph cycle detection results in a schedule for propagating outputs to inputs and invoking the FMU to produce new outputs.

To simulate a coupled system, the master algorithm of PyFMI proceeds by first providing inputs to all FMUs and then performing a step. This can be done for all models simultaneously and once all models have performed the step, information is exchanged between the FMUs so the next step can be computed.

The master algorithm supports error control, where the error estimate is based on Richardson extrapolation (Schierz et al. 2012). The error estimate is done by performing a global integration step twice using different inputs. A first step is performed using a step size of h. This step is compared with two steps of step size $h/2$, where inputs and outputs between the subsystems are updated before taking the second step of step size $h/2$.

Simulation of coupled systems using PyFMI are restricted to models which implement either the model exchange or the co-simulation API of FMI 2.0. The PyFMI master does not support coupled systems which include models with different APIs.

PyFMI supports the simulation of coupled systems that induce an algebraic loop. In this case, PyFMI uses the Tarjan's algorithm (Tarjan 1972) to compute the strongly connected components of the system, and then PyFMI solves such connected components simultaneously. The algebraic loop's support is limited to coupled systems formed with models that implement the model exchange API of FMI 2.0.

The version of the master which was tested at time of writing is included in JModelica.org 2.2 [7]. A detailed description of the master algorithm PyFMI can be found in Andersson (2016).

3. Applications

3.1. *PV Penetration Impact on Distribution Grid Voltage*

The distribution grid is including more and more DERs and active components such as PVs, smart inverters and EVs. To accurately simulate their impact on the distribution

[7] https://jmodelica.org/

grid we use the CyDER platform to couple a grid simulator (CYMDIST) with specific simulators representing PVs, EVs, and smart inverters. The benefit of this approach is to leverage state-of-the-art models such as for EV, or using vendor specific hardware, as opposed to the built-in simplified models embedded in power flow software such as CYMDIST.

In this use case, we have simulated multiple levels of PV penetration on the second half of a PG&E[8] utility feeder for two weeks in June 2016. We defined the second half of the feeder as the network beyond half the feeder's total cable distance. We run five use cases

(1) PV only use case, which is the base case
(2) PV plus reactive power compensation (injecting or absorbing reactive power)
(3) PV plus EV demand from a simulated profile using National Household Travel Survey data
(4) PV plus EV demand from power demand measures at a ChargePoint[9] station
(5) PV plus reactive power compensation plus EV demand from use case 4

The simulated feeder has a peak demand of 2.8 MW recorded in 2016 (figure 9). 50.7% of the customers are industrial, 45.4% are commercial facilities, while the remaining 3.9% are household customers. The feeder is 2.6 km long and includes capacitor banks with a nominal reactive power capacity of 300 kVAR, but no voltage regulator. The global horizontal irradiance used as input to the PV module is plotted on figure 7. PVs are homogeneously added on the second half of the feeder model (beyond 1.3 km from the feeder head), with a varying penetration level from 0% to 100%. The 100% penetration level (4.1 MW PV system) corresponds to a scenario where the energy produced by the PV systems over the course of 2016 is equal to the annual load energy demand.

In use case 2, each PV has a smart inverter (as defined in section 2.2.4) which is able to provide 30% of the PV rated capacity as reactive power. The EV load demands for use cases 3, 4 and 5 are plotted on figure 8. EV load demands are set to be less than half the feeder power demand at peak time, representing the consumption of approximately 1000 vehicles throughout a day. Figure 9 shows the net load at the feeder head for five days in June with six different PV penetrations. We observe that reverse power flows starts at an installed PV power of 1381 kW (33.3% penetration) and becomes larger than the load peak demand for an installed PV power of 4143 kW.

Figure 10 presents the highest simulated voltages as a function of the PV penetration level for each of the use case. The solid line represents the median for the first 2 weeks of June while the shaded area represents the 80% percentile. The co-simulation shows that in absence of controls, a total PV generation capacity of 830 kW (20% PV penetration) leads to over-voltages (voltage above 105% of the reference value).

In use case 2, smart inverters consume reactive power equal to 30% of the PV VA-rating at peak production if the voltage is above 107%, thus reducing over-voltage from 1.19 p.u. to 1.13 p.u. at 100% PV penetration.

In use cases 3 and 4, even though the EV load demand is uncontrolled, it still absorbs some of the PV active power, enabling PV penetration to 30% and 40%, respectively, without leading to over-voltages.

Use case 4 shows the importance of shifting the EV load demand in the afternoon in order to coincide with the peak PV generation. As the portion of EVs charging in the

[8] https://www.pge.com/
[9] https://www.chargepoint.com/

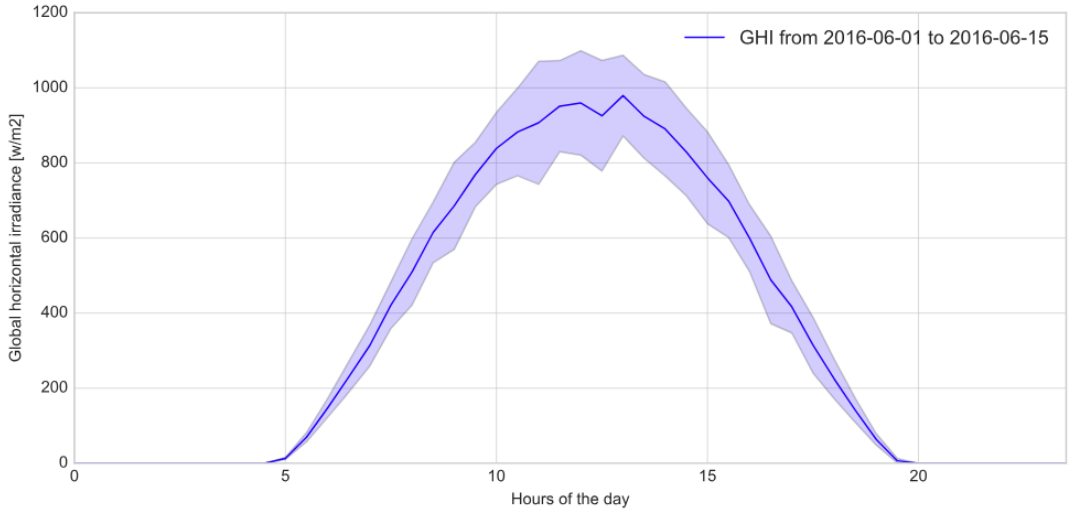

Figure 7. Global horizontal irradiance over the simulation period. The blue solid line corresponds to the median, and the shaded blue area to the standard deviation.

afternoon in use case 4 is greater than in use case 3, 10% more PVs can be installed.

In use case 5, the action of EVs and reactive power absorption enable to reduce the over-voltage from 1.19 p.u to 1.10 p.u at 100% of PV penetration.

Our analysis shows that the current PV hosting capacity of the studied feeder is around 20%, assuming today's load demand and no smart inverter capabilities. Reactive power compensation with PV inverter Volt/Var control reduces the worst-case over-voltages significantly, especially as the PV penetration approaches 100%. However, note that the increase in PV hosting capacity - while keeping the worst-case voltage below 105% of the reference - with reactive power compensation only is limited (around 2%). Nevertheless, together with a widespread deployment of EVs, PV inverter control can increase the PV hosting capacity to more than 40%. Future analysis will consider more feeder models, long-term scenarios for load demand evolution, as well as more sophisticated inverter control.

This application is an example of how a CyDER co-simulation allows multiple simulators to run together, each simulating a different aspect of the power system. This example includes a PV system, an EV charging station, a smart inverter, and the distribution grid. Leveraging the FMI standard for co-simulation allows to plug and play different simulators, unlocking the ability to represent the growing diversity of systems on the grid. For instance, the PV system simulator enables studying the tilt and orientation of PVs to reduce over-voltage issues. Furthermore, this application demonstrates the applicability of CyDER to analyze the impact of high penetration of DERs on the Grid.

3.2. *Hardware-in-the-loop*

The purpose of this Hardware-in-the-Loop (HIL) experiment is to test the developed FMUs in a real-world environment, which includes on-site PV generation, load modeling and simulation, and supervisory control. The demonstration is conducted at LBNLs newly built FLEXGRID, a facility for power systems HIL testing, which allows a broad range of test scenarios under emulated and real world conditions. It includes an Opal-RT real-time simulator as the center of operation, an Ametek 30

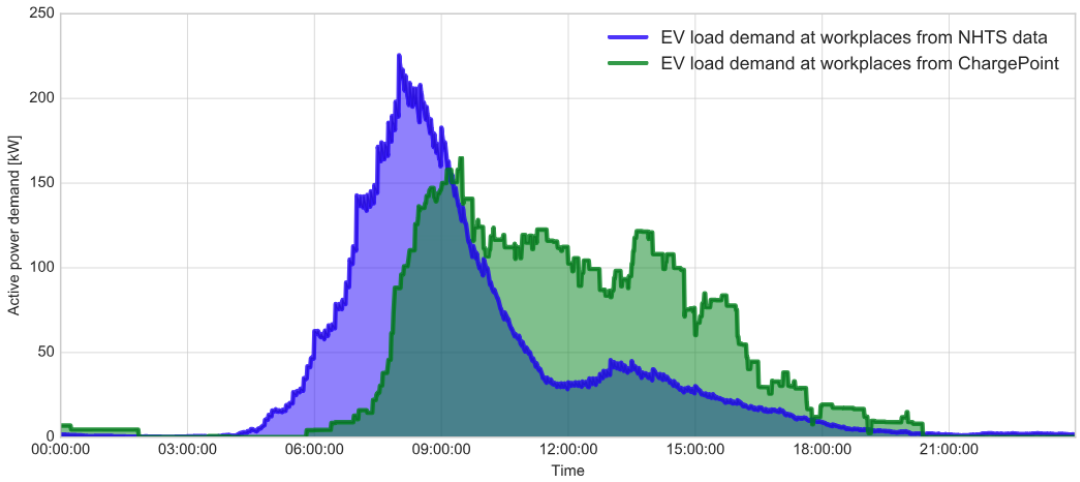

Figure 8. EV load demand for 100 vehicles from two different sources (scaled to 1000 EVs for this simulation). The blue curve represents EV load demand simulated using National Household Travel Survey (NHTS) data. The green curve represents EV load demand as recorded from a ChargePoint station.

kVA variable AC voltage source as a grid emulator, three single-phase SolarEdge PV inverter with a total of 15 kW PV and 19 kWh of storage, and an Ametek 3 kW load simulator. Power flow is measured by a high fidelity micro-synchrophasor measurement unit from Power Standards Lab. A simplified diagram of FLEXGRID is shown in Fig.11.

3.2.1. HIL Setup

A test scenario that includes multi-domain interaction is proposed. This scenario emulates a medium sized distribution grid, provided as a PG&E CYMDIST model, with extrapolated PV penetration at selected nodes. One selected node within the model is emulated by FLEXGRID, where nodal voltages are send and amplified by the grid emulator, and resulting PV generation is exposed to actual solar irradiations at the LBNL site in Berkeley, CA. A supervisory control is limiting PV active power generation or providing reactive power injection of all inverters in the model, based on emulated voltage measurements. This scenario, even though simplistic, demonstrates the ability of CyDER, as co-simulation, to simplify the workflow to set up a HIL test for controller field testing. Fig.12 shows the design of the described HIL setup.

At the center of the schematics is the real-time simulator Opal-RT. All components to the right are part of FLEXGRID, the ones to the left are part of CyDER. For CyDER purposes the Opal-RT simulator is encapsulated in a custom FMU, called Simulator API FMU, to allow simple utilization and configuration. The backend of the HIL simulation consists of two components, Opal RT, the real-time simulator, and the master PC. The real-time simulator exclusively communicates with the master PC to obtain simulation models and adapt parameters within the model. In the first step of initialization, the provided distribution model is loaded onto the simulator to set up and start the HIL emulation. The simulated PV inverters are controlled with an emulated Volt-Var control, running as a standalone Inverter Control FMU, whereas the actual inverters utilize the same FMU but parameterized as a Volt-Watt control. The communication between Inverter Control FMU and Simulator API FMU is through the FMI interface. The inverter FMU sets the inverter control based on measurements

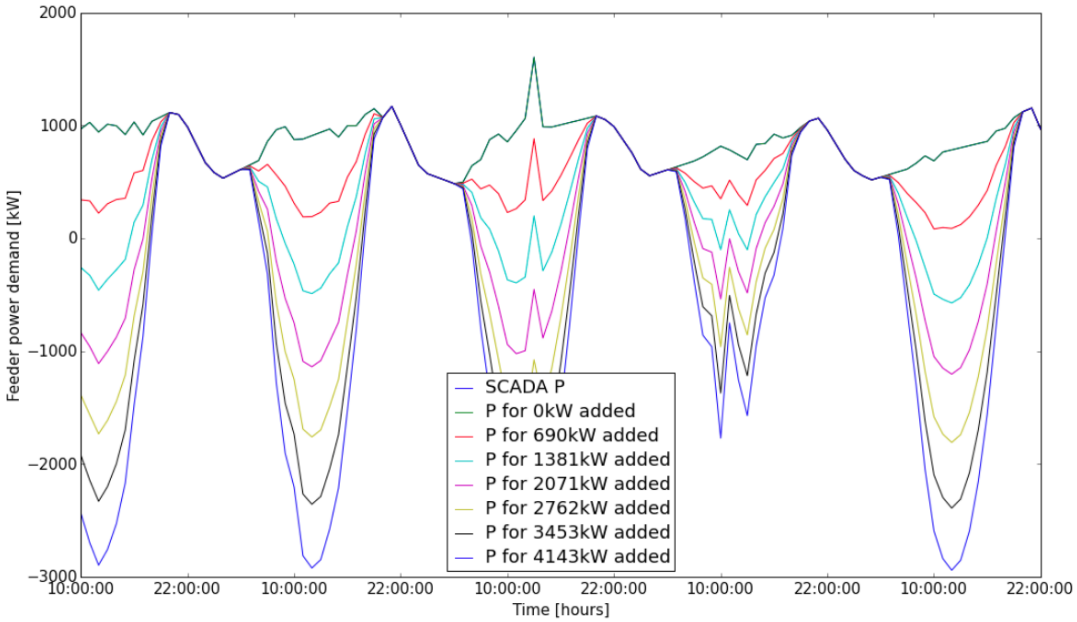

Figure 9. Feeder net load with various level of penetration of PVs. The blue curve is for the 4143 kW PV systems which represents the 100% PV penetration case.

acquired from the simulator.

3.2.2. Test Parameter

The FMUs for the simulated PV and inverter control are parameterized to reflect a high PV penetration scenario where high voltages may be apparent. All of the three simulated PV systems use TMY weather for San Francisco, and are orientated towards West, South, and East. The date in the weather file is chosen as one high generation day which matches the month of the actual test (June). The parameter for the simulated PVs are approximately 2 MW of rated peak power, efficiency of 12%, and tilt angle of 10 degree toward the orientation. The simulated PVs are connected in a symmetric delta connection, while the real PV system at FLEXGRID is subject to imbalances due to actual generation of the three independent PV systems. The real PV system is scaled from 15 kW up to 0.9 MW to force high voltages. The control system for the simulated PVs is a Volt-Var control with an output range from 0 to 100% of rated reactive power, which is 600 kVar in this case, whereas the real PV utilizes a Volt-Watt control with an output range from 100 to 0% of rated active power. The setpoints for both controller are set in accordance to PG&E, which is a deadband up to 1.033 Vp.u., and a output saturation at 1.07 Vp.u (PG&E 2017). The loads within the model are set based on a load allocation within the CYMDIST feeder model, where real SCADA data is used to allocate the demand, based on the power rating in CYMDIST. The allocated load profiles are then duplicated for each line, which triples the power flow, allowing for higher PV generation to ultimately force high voltages on the feeder. This step also allows the utilization of the feeder in a symmetric loading case. The date of the loads from SCADA data is again chosen to match the month of the test, and reflects a typical load profile.

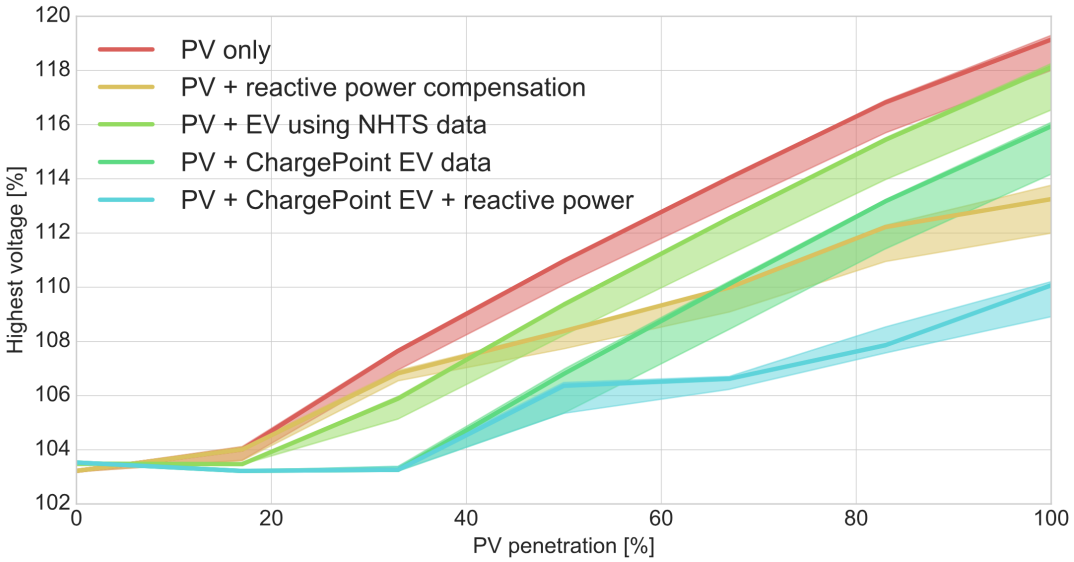

Figure 10. Result of the co-simulation where over-voltages (voltage level above 105% of the reference) are represented as function of the PV penetration.

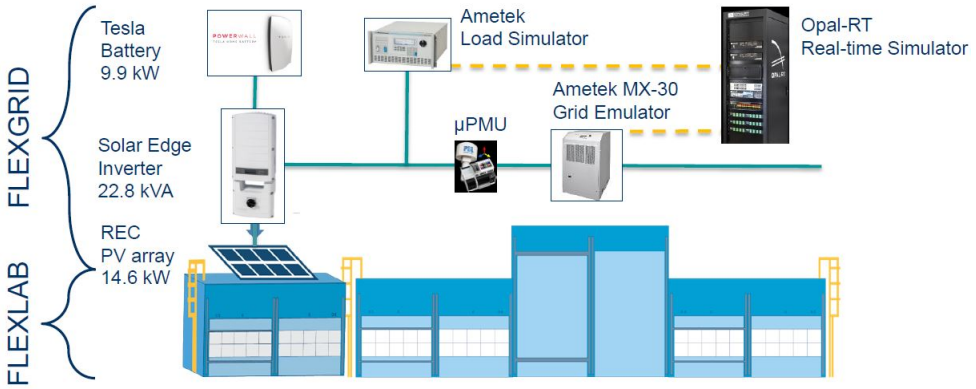

Figure 11. Overview of FLEXGRID, a HIL testing facility at LBNL.

3.2.3. HIL Results

The HIL test was executed for a full day of utilization (23 hours) with a co-simulation communication timestep of 5 minutes. While CyDER computes much faster in simulation mode, the addition of the real-time simulator FMU drastically limited the timestep within CyDER. Besides the 5 minute timestep for CyDER, the HIL test included a variety of sub-models which run with different timesteps. These include the real-time HIL model with a 20 micro-second timestep, the real-time grid model with a 10 milli-second timestep, and the load change within the model with a 15 minute timestep. The following plot shows the results of the test with power flow and control actions taken by CyDER.

The results in Fig.13 are a set of four time-series. The first plot shows the feeder base load, which is allocated from SCADA data, in purple, the simulated PV generation profiles for the orientations of East, South, and West in green, orange, and red respectively, and the real PV generation at FLEXGRID, in blue. The different timesteps of the system can be observed by the step function changes in the corresponding time-

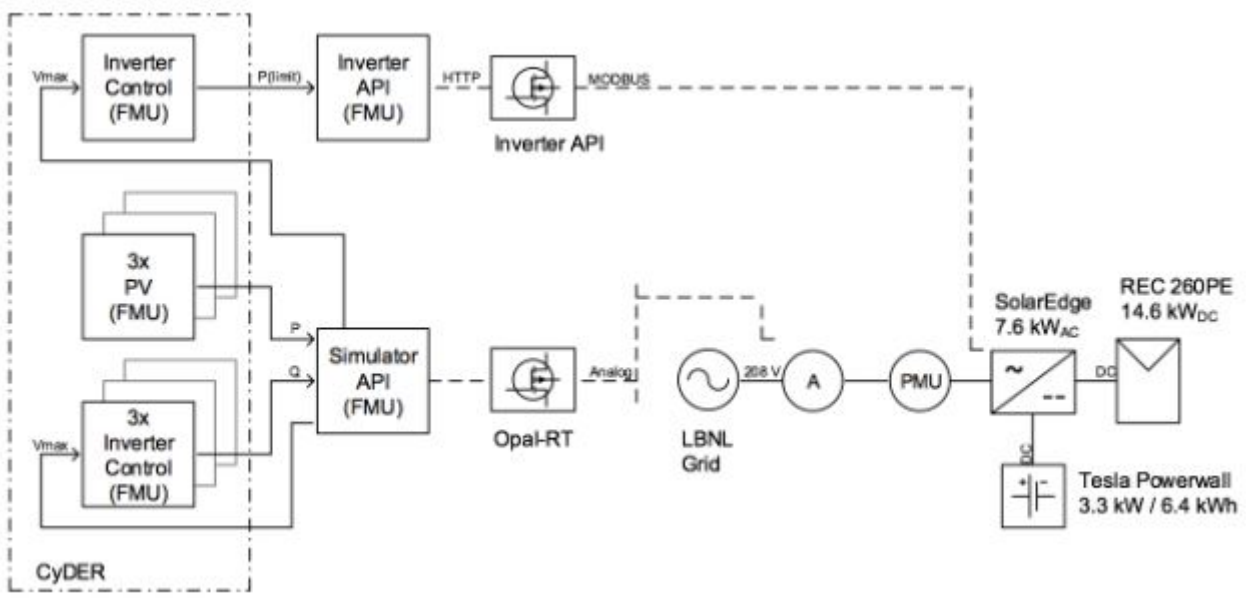

Figure 12. Test setup of CyDER for HIL.

series. The power flow from the various sources feeds the real-time simulation where the nodal voltages can be retrieved via the Simulator FMU, as shown in the second plot. These voltages are again subject to the 5 minute communication timestep. It can be seen that due to the scaling of the loads, the nodal voltages for the real PV and simulated PV2 are actually below the 0.95 Vp.u. threshold when no PV generation is present. However, once the sun rises the PV plants increase the nodal voltages by about 10%, which leads to high-voltage periods. For a period from 9 AM to 5 PM, the Volt-Var or Volt-Watt control take action by providing reactive power, or curtailing active power, in case of the real PV system. The control signals can be observed in the third plot. Depending on the location of the node along the feeder, and power demand of surrounding loads, the control signals differ. While the simulated PV2 system provides reactive power all day, due to the high nodal voltages, the simulated PV1 does not provide any reactive power. The real PV system was controlled down to around 50%, which actually curtailed generation during the peak hours. The last plot shows the feeder load at the head as an aggregation of all loads and generation on the feeder. It can be seen that from 9 AM to 5 PM the power flow reverses is in the order of double the base load. Roughly speaking it can be stated that the test was conducted with a 200% PV penetration scenario.

3.2.4. Discussion

The feeder model for the HIL test provided by PG&E turned out to include a lot of redundancy and safety, and it required several upscale efforts to result in high- and over-voltage scenarios where the developed controls could be tested. Interesting hereby is that even with a peak PV generation of approximately 200% of the base load (in terms of instantaneous power), the voltages just passed the high-voltage threshold.

The model accuracy of the real-time model was granted sufficient for the current application, but further improvement could be achieved by reducing the differences between the ePhasorsim and CYMDIST implementation of balanced power flow with

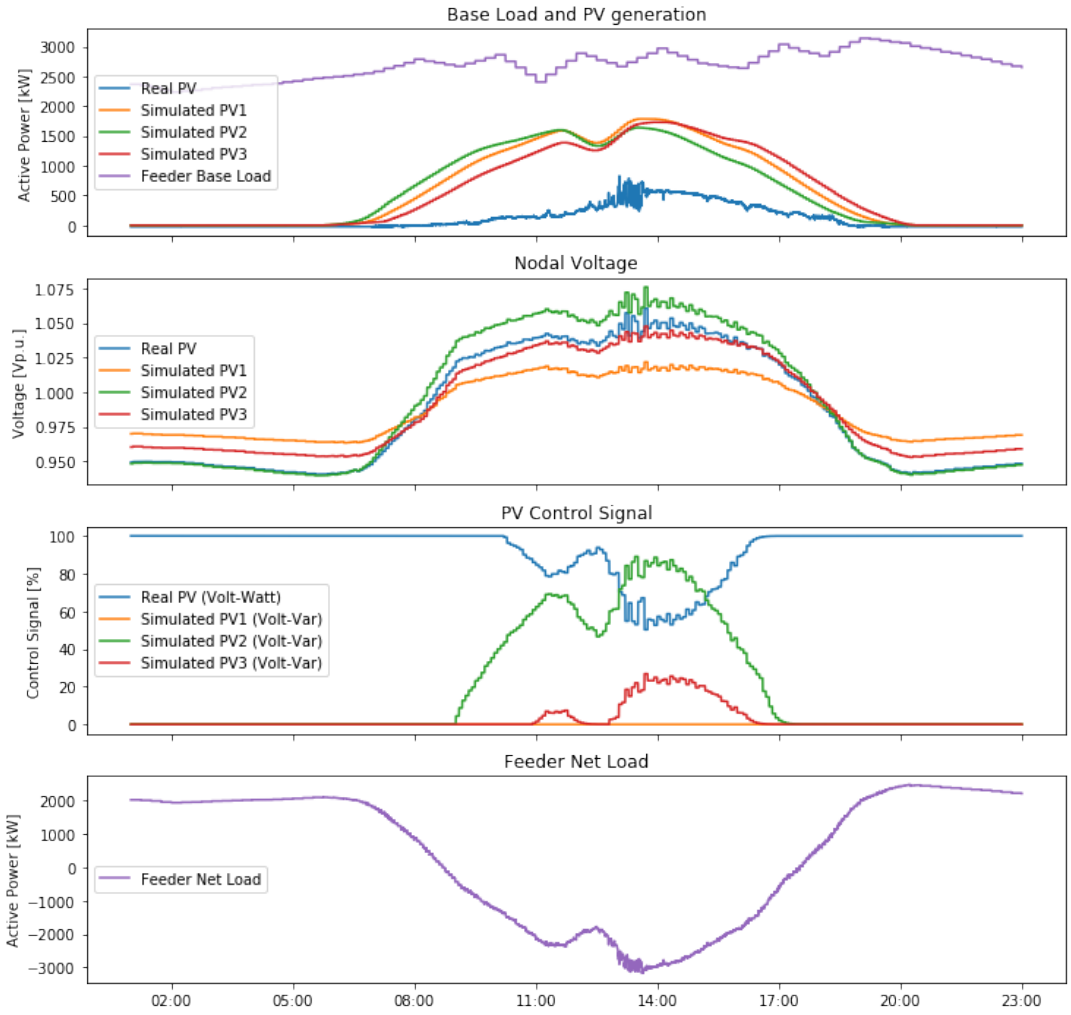

Figure 13. Results of HIL test of CyDER at FLEXGRID.

single-phase loads. Another contributor to the discrepancies in resolution can be found by the limited timestep and data acquisition. The contributing factor hereby is the Python API of the Opal-RT simulator, which is relatively slow in execution, and currently only allows approximately 250 queries at a model instance. If the number of queries is exceeded, then the connection to the real-time model breaks, ultimately terminating the HIL test. The timestep was therefore adjusted to safely cover one full day of operation.

4. Conclusion and Future Work

This paper presented CyDER, a co-simulation platform which uses the open industry standard FMI to link simulation models for grid planning and operation. The use of CyDER was demonstrated in two applications, where the results showed that appropriate controls help mitigate potential grid instability as a consequence of high DERs. To support the export of existing and new simulation tools as FMUs, a Python utility has been developed. Future work will include extensions of the python utility to sup-

port the export of server-based simulation modules. This will further accelerate the adoption of FMI in power systems and increase the number of simulation tools which support the FMI interface. Further expansion of the framework might include an optimization module or FMU which can then be used to utilize CyDER for planning and operational purposes. Examples hereby would be the optimal placement of a finite amount of μPMU sensors along the feeder, optimal placement and setpoints of DERs, or other supervisory predictive control decisions. Furthermore, the PV penetration analysis presented in this paper will be extended with consideration of long-term load demand scenarios and integration of predictive controls for PV and battery inverters. Last, in addition to the current quasi steady-state simulations, future efforts will focus on dynamic and transient power system simulations using CyDER.

The FMI standard is well positioned to become the de facto standard for co-simulation and model exchange which will increase re-usability, and interoperability of simulation tools. However, better support for variable types such as vectors or structures would be desirable to facilitate large-scale use cases. For instance, the current form of the FMI standard does not allow to pass a vector of input values to an FMU, and the user is limited to scalar variable inputs and outputs. As power systems potentially have thousand of inputs and outputs, this restricts usability. As of today, the FMI standards committee convened a working group to add support for vectors to the FMI standard, which will improve scalability and usability of the CyDER platform.

Acknowledgment

This work is supported by the U.S. Department of Energy SunShot National Laboratory Multiyear Partnership (SuNLaMP) program award number 31266.

References

Amarasekara, B. Ranaweera, C., Nirmalathas, A., and Evans, R. “Co-simulation platform for smart grid applications.“ 2015 IEEE Innovative Smart Grid Technologies - Asia (ISGT ASIA), Bangkok, 2015. pp. 1-6. doi: 10.1109/ISGT-Asia.2015.7387091.

Andersson, C. “Methods and Tools for Co-Simulation of Dynamic Systems with the Functional Mock-up Interface“. PhD Thesis, Lund University, Sweden.

Blochwitz et al. “Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models“. Proceedings of the 9th International Modelica Conference , Munich, Germany, 2012. pp. 173-184.

Buescher et al. “Integrated Smart Grid simulations for generic automation architectures with RT-LAB and mosaik“. 2014 IEEE International Conference on Smart Grid Communications (SmartGridComm). Venice, 2014, pp. 194-199. doi: 10.1109/SmartGridComm.2014.7007645.

California Distribution Resources Plan (R.14-08-013). Integration Capacity Analysis Working Group, Final Report, 2016.

Chatzivasileiadis et al., “CyberPhysical Modeling of Distributed Resources for Distribution System Operations“. In Proceedings of the IEEE, vol. 104, no. 4, pp. 789-806, April 2016. doi: 10.1109/JPROC.2016.2520738

Chombard et al. “MODELISAR From System Modeling to S/W running on the Vehicle“. https://itea3.org/project/modelisar.html

Daily et al. “ FNCS: A Framework for power system and communication networks co-simulation“. In Proceedings of the Symposium on Theory of Modeling & Simulation - DEVS Integrative, Tampa, Florida, 2014, pp.36:1-36:8.

Godfrey et al. “Modeling Smart Grid Applications with Co-Simulation“. 2010 First IEEE International Conference on Smart Grid Communications, Gaithersburg, MD, 2010, pp. 291-296. doi: 10.1109/SMARTGRID.2010.5622057.

Gomes et al. “Co-simulation: State of the art“. 2017.

Gomez-gualdron et al. “Simulating a Multi-Agent based Self-Reconfigurable Electric Power Distribution System“. 2006 IEEE Workshops on Computers in Power Electronics, Troy, NY, 2006, pp. 1-7. doi: 10.1109/COMPEL.2006.305644.

Hadjsaid, Nouredine and Sabonnadiere, Jean-Claude. “Smart power grids“, 2012, ISBN 978-1-84821-261-9.

Nouidui, T. and Wetter, M. “SimulatorToFMU: A Python Utility to Support Building Simulation Tool Interoperability“, To appear in Proceedings of the 2018 Building Performance Analysis Conference and SimBuild, Chicago, IL.

Pacific Gas and Electric “Distribution Interconnection Handbook 2017“, available at https://www.pge.com/includes/docs/pdfs/shared/customerservice/nonpgeutility/electrictransmission/handbook/distribution-interconnection-handbook.pdf

Palmintier et al. “IGMS: An Integrated ISO-to-Appliance Scale Grid Modeling System“. In IEEE Transactions on Smart Grid, vol. 8, no. 3, pp. 1525-1534, May 2017. doi: 10.1109/TSG.2016.2604239.

Roche et al. “A Framework for Co-simulation of AI Tools with Power Systems Analysis Software“. 2012 23rd International Workshop on Database and Expert Systems Applications, Vienna, 2012, pp. 350-354. doi: 10.1109/DEXA.2012.9.

Sandia National Laboratories, “SCEPTRE“. https://www.osti.gov/servlets/purl/1376989

Saurabh et al. “A System-of-Systems Approach for Integrated Energy Systems Modeling and Simulation“, In Proceedings of the Conference on Summer Computer Simulation, Chicago, IL.

Saxena et al. “Quantifying EV battery end-of-life through analysis of travel needs with vehicle powertrain models“. Journal of Power Sources Vol. 242, pp. 265-276.

Schierz et al. “Co-simulation with communication step size control in an FMI compatible master algorithm“. In Proc. of the 9th International Modelica Conference, Munich, Germany, pp, 205-214.

Solar Energy Industries Association, Solar Industry Data, 2017

Tarjan, R. “Depth-first search and linear graph algorithms“. SIAM Journal of Computing Vol.1 (2), pp. 146-160.

Top, P. “GridDyn User Manual“, August 2016.

ZeroMQ Available at http://zeromq.org

Zeigler, B., Vahie, S. “DEVS formalism and methodology: unity of conception/diversity of application“. In Proceedings of the 25th conference on Winter simulation, Los Angeles, CA.