Cyber-Physical Co-simulation Framework between Typhon HIL and OpenDSS for Real-Time Applications

Raju Wagle[1*], Pawan Sharma[1], Mohammad Amin[2] and Francisco Gonzalez-Longatt[3]

[1]Department of Electrical Engineering, UiT The Arctic University of Norway, Narvik, Norway
[2]Department of Electric Power Engineering, NTNU, Norway
[3]Centre for Smart Grids, University of Exeter, Exeter, United Kingdom

raju.wagle@uit.no
pawan.sharma@uit.no
mohammad.amin@ntnu.no
fglongatt@fglongatt.org

Abstract. Cyber infrastructures have been extensively used for power system monitoring, control, and operation because of the development of new information and communications technology (ICT) in power systems. Deployment of cyber-physical co-simulation in the case of a realistic distribution network is still a big challenge. Hence, to solve the problem of modelling complex distribution networks, a cyber-physical co-simulation framework is proposed in this chapter. The proposed framework consists of a cybernetic layer, a physical layer, and a co-simulation framework between OpenDSS and Typhoon HIL. The cyber layer consists of software and tools to model the distribution system and communicate with the physical layer. The physical layer is the Typhoon HIL real-time simulator consisting of virtual or real controllable devices. A realistic framework to execute the real-time simulation using the Typhoon HIL SCADA system and Python-based co-simulation is created in this chapter. The real-time simulation demonstrates the proposed framework's effectiveness in observing the distribution network's voltage profile due to real-time variation in reactive power from the PV.

Keywords: Cyber-physical testbed, Co-simulation, Typhoon HIL, OpenDSS, Python, Real-time simulation.

1 Introduction

With the growing integration of converters-based renewable energy sources with advanced ICT (information and communication technology) in the electrical distribution network, the distribution system operation demands more robust and adequate monitoring and control infrastructure. Cyber-physical co-simulation is one of the suitable strategies for managing this complexity [1]. The significance of co-simulation in optimal reactive power control is highlighted in [2]. Cyber-physical co-simulation frameworks integrate the physical and digital or cybernetic systems to raise the standard of monitoring and control of the physical system by harnessing the powers of the cyber system. In addition, co-simulation is the synchronized use of two or even more simulation models that have different operating conditions [3]. Numerous tasks such as sensing, calculation, communication, and actuation are included in many applications like industrial automation, aeronautical and automotive control, power grid management, and inaccessible and dangerous investigations using a cyber-physical co-simulation framework [4].

Research on cyber-physical co-simulation is currently continuing in many fields, including the automobile industry, the military, building science, and energy. As a result, there are currently several general co-simulation systems available, each with unique features and levels of usefulness. Even though cyber-physical co-simulation is growing, there are still several problems that need to be resolved, especially while developing a framework for modelling the distribution network. A real-time cyber-physical test bed for microgrid control is proposed in [5], where the power system network is designed in RSCAD, and the real-time digital simulator (RTDS) is used

for real-time co-simulation. Similarly, in [6], a real-time simulation test bed that operates between OpalRT and Matlab Simulink is proposed. Real-Time Laboratory (RT-LAB), and the communication network simulation developed with OPNET are observed in [7]. A cyber-physical testbed for co-simulation for real-time testbed for reactive power control in smart inverter is proposed in [8]. A detailed review of the real-time co-simulation testbed is studied in [9]. In most of the previous studies, a cyber-physical co-simulation testbed is proposed for microgrids or small distribution networks. However, to analyze a real distribution network, the distribution network solver considered in the co-simulation testbed should be able to solve all types of distribution networks. Also, the real-time operation in the case of distribution networks needs to be robust and fast while solving the complex distribution network. OpenDSS is a very powerful tool specially designed to solve the distribution network. Hence, this work intends to develop a cyber-physical co-simulation framework to run the real-time simulation between the OpenDSS and Typhoon HIL.

The chapter presents a stronger emphasis on standards and common software tools, as well as the application of these tools to formulate a cyber-physical co-simulation framework between the Typhoon HIL and OpenDSS. A script to create communication is attached with a short explanation to achieve the main goal of the cyber-physical co-simulation.

2 Development of Cyber-Physical Co-Simulation Framework between Typhoon HIL and OpenDSS

The proposed cyber-physical co-simulation framework is depicted in Fig. 1, and it consists of two layers: (a) the Cybernetic layer and (b) the physical layer (top and bottom of Fig. 1, respectively). A classic PC, called workstation, uses the local CPU to run all the pieces of software of the cybernetic layer. It includes software to define and control de interactions between the layers and the distribution system solver. Finally, the physical layer includes the Typhoon HIL real-time simulator in this case.

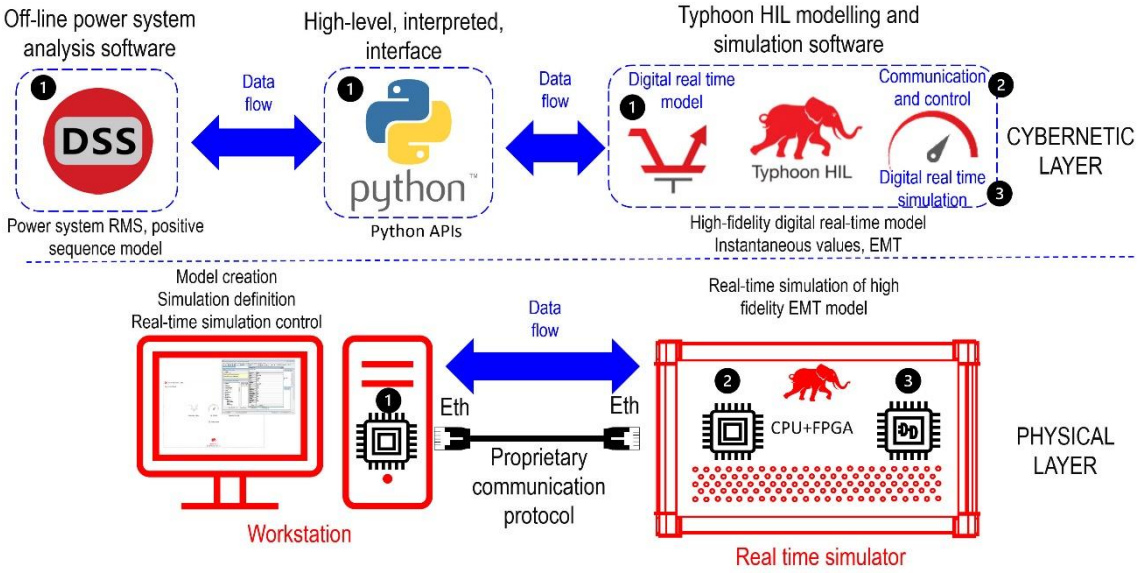

Fig. 1 Framework for cyber-physical co-simulation framework using OpenDSS and the Typhoon HIL real-time modelling and simulation framework

Development of Typhoon HIL and OpenDSS co-simulation scenarios requires a significant amount of time, especially when dealing with multiple numbers of distributed resources.

To develop a co-simulation framework three main tasks should be achieved:

(i) Create an appropriate models of a distribution network and distributed resources in power system analysis softwarte, in this case OpenDSS,

(ii) Builid and appropriate system to control and communicate with the real-time model and simulation, using Typhoon HIL Schematic and SCADA.

(iii) Create a a suitable python program to exchange information between OpenDSS and Typhoon HIL and vice versa.

The first task is creating a power network model, in this case, a distribution system; it includes the information of connected loads and PVs in OpenDSS, which is used as a base model for further studies. The size, location, and a number of PVs interconnection depend on different factors; however, in this study, the PVs and load are connected as per the test study created by CIGRE. The second task involves designing the typhoon HIL schematic editor and the typhoon HIL SCADA to exchange information between the Typhoon HIL real-time simulator and the Typhoon HIL SCADA. In the schematic editor, a subsystem is created to get the information from SCADA and to send the information to Typhoon HIL. The number of subsystems depends on the number of devices to be controlled from the SCADA terminal. The Typhoon HIL schematic editor has an abundant number of libraries to create an interface dependent on the requirement. In the third task, a python program must be created to initialize, execute the overall process, get the information from the real-time simulator, and send the signal to the real-time simulator. In this chapter, the considered distributed network is a European medium voltage network created by CIGRE Task Force C6.04 presented in the report " *Benchmark Systems for Network Integration of Renewable and Distributed Energy Resources*"[10]. In this chapter only PVs are considered as the distributed resources.

2.1 Creating models of distribution networks and distributed resources in OpenDSS

The Open Distribution System Simulator (OpenDSS) is a simulation tool for modelling electrical distribution systems. It supports all types of steady-state analyses in distribution systems. The most popular application of OpenDSS is Distribution system analysis and planning, Multiphase power flow analysis, interconnection studies of Distribution generations, Harmonics, and inter-harmonics analysis. Different modes of operation like snapshot power flow, time series power flow, Harmonic study, Fault analysis, etc. can be achieved by using OpenDSS.

The overall structure of the OpenDSS [11] is shown in Fig. *2*

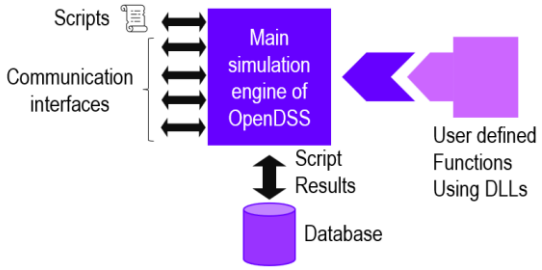

Fig. 2 OpenDSS database and communication interfaces and mechanisms.

A distribution network might require several types of devices, such as distributed generators, transformers, lines, and loads.
Complete detail of the network components is required to model a distribution network appropriately. Modelling details of power components models and the process of creating the power network models in OpenDSS can be found in the OpenDSS manual [12].
This subsection presents an explanation of the detailed modelling specific to the power network considered in this chapter. To model a network with thorough knowledge of the network components, the following steps, described in the subsequent subsection, are followed to create a distribution network.

2.1.1 **Creating a network as a circuit**

The first step in creating a model in OpenDSS is defining a new circuit. When a new

circuit is created, it is installed as a three-phase voltage source named "Source" connected to a bus named "SourceBus" with a reasonable short circuit strength. The new circuit has many parameters that users can define or set to default values. To create the European MV CIGRE network [13] the default values are replaced by the ones provided in CIGRE. Then, in the script of OpenDSS, the new circuit is defined, as shown in the code below.

```
// creating the slack bus and the main circuit
New Circuit.EuropeanMV
~ basekv=110 pu=1.0 phases=3 bus1=SourceBus
~ R0=1e-6 R1=1e-6
~ X0=1e-6 X1=1e-6
```

2.1.2 **Creating Transformers**

Transformers are one of the important components in an electric distribution network. To model a transformer in OpenDSS, the following script can be used.

```
//SUB TRANSFORMER DEFINITION
New Transformer.TR1 Phases=3 R=0.001 XHL=1.92
~ wdg=1 bus=SourceBus kV=110 kVA=25000 conn=Delta Numtaps=8
Mintap=0.9 Maxtap=1.1
~ wdg=2 bus=bus1 kV=20 kVA=25000 conn=Wye Numtaps=32 Mintap=0.9
Maxtap=1.1 Tap=1.03125 !1.1-11*0.00625

New Transformer.TR2 Phases=3 R=0.001 XHL=1.92
~ wdg=1 bus=SourceBus kV=110 kVA=25000 conn=Delta Numtaps=8
Mintap=0.9 Maxtap=1.1
~ wdg=2 bus=bus12 kV=20 kVA=25000 conn=Wye Numtaps=32 Mintap=0.9
Maxtap=1.1 Tap = 1.03125 !1.1-11*0.00625
```

In this model, there are two distribution transformers connected to the grid. It is considered that each transformer has $\pm 10\%$ load changing taps, with each tap change corresponding to $\pm 0.625\%$. However, in the transformer used here, the taps are fixed.

2.1.3 **Creating lines**

The lines of the network are created as an object that connects two buses in the network. The line parameters are represented by the line codes, which can be defined separately in the script of OpenDSS. The code for creating line codes and lines in OpenDSS is given below. The data are modified as per our requirement.

```
// creating line codes for the lines
//considering single phase values with C
New LineCode.1 nphases=3 R1=0.501 X1=0.716 R0=0.817 X0=1.598
C1=151.24 C0=151.24 Units=km
New LineCode.2 nphases=3 R1=0.501 X1=0.366 R0=0.658 X0=1.611
C1=10.101 C0=4.0764 Units=km
// creating the lines for the network
New Line.LINE1 Bus1=bus1 Bus2=bus2 phases=3 LineCode=1
Length=2.82 Units=km
New Line.LINE2 Bus1=bus2 Bus2=bus3 phases=3 LineCode=1
Length=4.42 Units=km
New Line.LINE3 Bus1=bus3 Bus2=bus4 phases=3 LineCode=1
Length=0.61 Units=km
New Line.LINE4 Bus1=bus4 Bus2=bus5 phases=3 LineCode=1
Length=0.56 Units=km
New Line.LINE5 Bus1=bus5 Bus2=bus6 phases=3 LineCode=1
Length=1.54 Units=km
New Line.LINE6 Bus1=bus6 Bus2=bus7 phases=3 LineCode=1
Length=0.24 Units=km
```

```
New Line.LINE7 Bus1=bus7 Bus2=bus8 phases=3 LineCode=1
Length=1.67 Units=km
New Line.LINE8 Bus1=bus8 Bus2=bus9 phases=3 LineCode=1
Length=0.32 Units=km
New Line.LINE9 Bus1=bus9 Bus2=bus10 phases=3 LineCode=1
Length=0.77 Units=km
New Line.LINE10 Bus1=bus10 Bus2=bus11 phases=3 LineCode=1
Length=0.33 Units=km
New Line.LINE11 Bus1=bus11 Bus2=bus4 phases=3 LineCode=1
Length=0.49 Units=km
New Line.LINE12 Bus1=bus3 Bus2=bus8 phases=3 LineCode=1
Length=1.30 Units=km
New Line.LINE13 Bus1=bus12 Bus2=bus13 phases=3 LineCode=2
Length=4.89 Units=km
New Line.LINE14 Bus1=bus13 Bus2=bus14 phases=3 LineCode=2
Length=2.99 Units=km
New Line.LINE15 Bus1=bus14 Bus2=bus8 phases=3 LineCode=2
Length=2.0 Units=km
```

2.1.4 **Creating loads**

Loads can be defined by a specific piece of script in OpenDSS. The loads can be modelled as static loads or time-dependent loads. Also, they can also be defined as single-phase loads or three-phase loads. In the code below, the modelling of the load is done. In this study, the loads are modelled as static 3-phase loads.

```
//creating residential load for the networks
New Load.LOAD1 Phases=3 Bus1=bus1 kV=20 kva=15300 PF=0.98
New Load.LOAD2 Phases=3 Bus1=bus3 kV=20 kva=285 PF=0.97
New Load.LOAD3 Phases=3 Bus1=bus4 kV=20 kva=445 PF=0.97
New Load.LOAD4 Phases=3 Bus1=bus5 kV=20 kva=750 PF=0.97
New Load.LOAD5 Phases=3 Bus1=bus6 kV=20 kva=565 PF=0.97
New Load.LOAD6 Phases=3 Bus1=bus8 kV=20 kva=605 PF=0.97
New Load.LOAD7 Phases=3 Bus1=bus10 kV=20 kva=490 PF=0.97
New Load.LOAD8 Phases=3 Bus1=bus11 kV=20 kva=340 PF=0.97
New Load.LOAD9 Phases=3 Bus1=bus12 kV=20 kva=15300 PF=0.98
New Load.LOAD10 Phases=3 Bus1=bus14 kV=20 kva=215 PF=0.97

//creating commercial load for the networks
New Load.LOAD11 Phases=3 Bus1=bus1 kV=20 kva=5100 PF=0.95
New Load.LOAD12 Phases=3 Bus1=bus3 kV=20 kva=265 PF=0.85
New Load.LOAD13 Phases=3 Bus1=bus7 kV=20 kva=90 PF=0.85
New Load.LOAD14 Phases=3 Bus1=bus9 kV=20 kva=675 PF=0.85
New Load.LOAD15 Phases=3 Bus1=bus10 kV=20 kva=80 PF=0.85
New Load.LOAD16 Phases=3 Bus1=bus12 kV=20 kva=5280 PF=0.95
New Load.LOAD17 Phases=3 Bus1=bus13 kV=20 kva=40 PF=0.85
New Load.LOAD18 Phases=3 Bus1=bus14 kV=20 kva=390 PF=0.85
```

2.1.5 **Creating PV systems**

PV systems in OpenDSS can be modelled in several ways. In some studies, a user-defined control application is needed. In such cases, the user may model the PV system as a synchronous generator. However, in this study, the PV systems are modelled as defined in the manual of OpenDSS. The code below includes the definition of the PV systems considered in this study. The required number of PVs is created as per the simulation studies considered.

```
// creating PV in the network
// P-T curve is per unit of rated Pmpp vs temperature
// This one is for a Pmpp stated at 25 deg
New XYCurve.MyPvsT npts=4  xarray=[0  25  75  100]
yarray=[1.2 1.0 0.8  0.6]
```

```
// efficiency curve is per unit eff vs per unit power
New XYCurve.MyEff npts=4  xarray=[.1  .2  .4  1.0]
yarray=[.86  .9  .93  .97]
// per unit irradiance curve (per unit if "irradiance"
property)
New Loadshape.MyIrrad npts=24 interval=1
~mult=[0 0 0 0 0 0 .1 .2 .3  .5  .8  .9  1.0  1.0  .99
.9  .7  .4  .1 0  0  0  0  0]
// 24-hr temp shape curve
New Tshape.MyTemp npts=24 interval=1
~temp=[25, 25, 25, 25, 25, 25, 25, 25, 35, 40, 45, 50  60
60  55 40  35  30  25 25 25 25 25 25]

// pv definition at  bus 3 of size 25 kW
New PVSystem.PV1 phases=3 bus1=bus3 kV=20  kVA=25
irrad=1
~Pmpp=20 temperature=25 PF=1 %cutin=0.1 %cutout=0.1
~ effcurve=Myeff  P-TCurve=MyPvsT Daily=MyIrrad
TDaily=MyTemp
// pv definition at  bus 4 of size 25 kW
New PVSystem.PV2 phases=3 bus1=bus4 kV=20  kVA=25
irrad=1
~Pmpp=20 temperature=25 PF=1 %cutin=0.1 %cutout=0.1
~ effcurve=Myeff  P-TCurve=MyPvsT Daily=MyIrrad
TDaily=MyTemp
// pv definition at  bus 5 of size 30 kW
New PVSystem.PV3 phases=3 bus1=bus5 kV=20  kVA=30
irrad=1
~Pmpp=30 temperature=25 PF=1 %cutin=0.1 %cutout=0.1
~ effcurve=Myeff  P-TCurve=MyPvsT Daily=MyIrrad
TDaily=MyTemp
// pv definition at  bus 6 of size 30 kW
New PVSystem.PV4 phases=3 bus1=bus6 kV=20  kVA=30
irrad=1
~Pmpp=30 temperature=25 PF=1 %cutin=0.1 %cutout=0.1
~ effcurve=Myeff  P-TCurve=MyPvsT Daily=MyIrrad
TDaily=MyTemp
// pv definition at  bus 8 of size 30 kW
New PVSystem.PV5 phases=3 bus1=bus8 kV=20  kVA=30
irrad=1
~Pmpp=30 temperature=25 PF=1 %cutin=0.1 %cutout=0.1
~ effcurve=Myeff  P-TCurve=MyPvsT Daily=MyIrrad
TDaily=MyTemp
// pv definition at  bus 9 of size 30 kW
New PVSystem.PV6 phases=3 bus1=bus9 kV=20  kVA=30
irrad=1
~Pmpp=30 temperature=25 PF=1 %cutin=0.1 %cutout=0.1
~ effcurve=Myeff  P-TCurve=MyPvsT Daily=MyIrrad
TDaily=MyTemp
// pv definition at  bus 10 of size 40 kW
New PVSystem.PV7 phases=3 bus1=bus10 kV=20  kVA=40
irrad=1
~Pmpp=40 temperature=25 PF=1 %cutin=0.1 %cutout=0.1
~ effcurve=Myeff  P-TCurve=MyPvsT Daily=MyIrrad
TDaily=MyTemp
// pv definition at  bus 11 of size 10 kW
New PVSystem.PV8 phases=3 bus1=bus11 kV=20  kVA=10
irrad=1
~Pmpp=10 temperature=25 PF=1 %cutin=0.1 %cutout=0.1
```

```
~ effcurve=Myeff  P-TCurve=MyPvsT Daily=MyIrrad
TDaily=MyTemp
// this PV is the one that replaces wind turbine of size
1500 KW in original paper
// pv definition at  bus 7 of size 1500 kW
New PVSystem.PV9 phases=3 bus1=bus7 kV=20  kVA=1500
irrad=1
~Pmpp=1500 temperature=25 PF=1 %cutin=0.1 %cutout=0.1
~ effcurve=Myeff  P-TCurve=MyPvsT Daily=MyIrrad
TDaily=MyTemp
```

2.1 Creating Typhoon HIL Schematic and SCADA

The important part of real-time co-simulation using OpenDSS and Typhoon HIL is the creation of the Typhoon HIL schematic and the SCADA. In this subsection, detailed steps to create a schematic and SCADA are presented.

2.1.1 **Creation of Typhoon HIL Schematic**

Typhoon HIL schematic editor is a graphical user interface-based modelling system. The model in the schematic works as a bridge between the Typhoon HIL and the OpenDSS simulator. Variables are controlled in real-time in the Typhoon HIL SCADA. An appropriate definition of the controlled variables is performed in the Typhoon HIL Schematic. Finally, the controlled variables are passed to OpenDSS by using the Python API in the HIL SCADA. This process continues with the width of the simulation. Fig. 3 shows a schematic to communicate between Typhoon HIL and OpenDSS. In the figure, the inputs and the outputs blocks are the signal processing block to receive and send the signal.

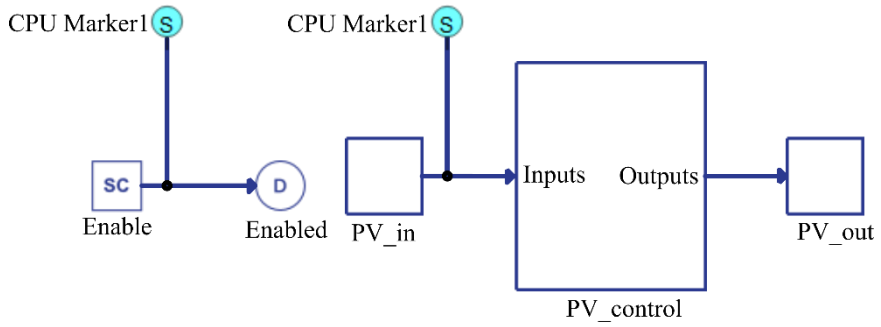

Fig. 3 Schematic diagram presenting the control of a PV distributed generator.

2.1.1 **Creation of Typhoon HIL SCADA**

Typhoon HIL SCADA is essential in realizing the real-time co-simulation between Typhoon HIL and OpenDSS. SCADA contains different panels to initialize co-simulation, control, and monitor expected outputs. The beautiful part of designing SCADA is that the Python program can be utilized effectively. Three important factors to be considered while designing the Typhoon HIL SCADA for co-simulation are listed below.

- Initializing Co-Simulation framework between Typhoon HIL and OpenDSS
- Development of panel to execute real-time control
- Development of a panel for monitoring the output in a real-time framework

2.1.1.1 **Initializing co-simulation framework between Typhoon HIL and OpenDSS**

To initiate the co-simulation between Typhoon HIL and OpenDSS, first, the OpenDSS shared library is interfaced using the Python library. OpenDSSDirect.py is the python interpreter to interact with Python from Typhoon HIL. OpenDSSDirect.py is imported in Typhoon HIL SCADA to execute the desired work. More details on how to import Python-based modules to Typhoon HIL python core can be found in Typhoon HIL operating manual [14]. Second, a program to communicate with the OpenDSS on startup and during the real-time simulation needs to be scripted in Python. Fig. 4 shows the panel developed in Typhoon HIL SCADA to initiate co-simulation. Different macro

widgets can be placed in a panel to define programs for initializing co-simulation.

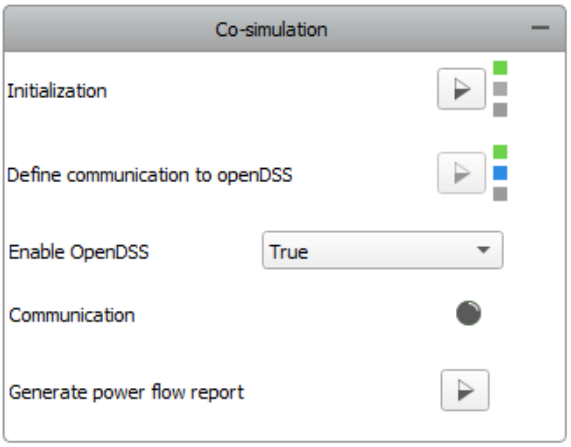

Fig. 4 Co-simulation setup panel.

To initialize the real-time co-simulation framework, the following code is scripted to initialize the OpenDSS parameters in Typhoon HIL SCADA.

```
# NOTE: Variables and functions defined here will be
# Available for use in all Macro and Expression scripts.
# NOTE: This code is always executed prior simulation
start.

# Variable 'SETTINGS_DIR' holds directory where loaded
Panel .cus file is located.
# Also you can call 'get_settings_dir_path()' function in
any
# Macro and Expression scripts to get the same directory.
SETTINGS_DIR = get_settings_dir_path()

# The 'add_to_python_path(folder)' function can be used to
add custom folder
# with Python files and packages to the PYTHONPATH. After
folder is added, all Python
# files and Python packages from it can be imported into
the SCADA Namespace.

# HIL API is imported as 'hil'
# SCADA API is imported as 'panel'
# SCADA API constants are imported as 'api_const'
# Numpy module is imported as 'np'
# Scipy module is imported as 'sp'
# Schematic Editor model namespace is imported as 'scm'
# Function for printing to HIL SCADA Message log is imported
as 'printf'.

import os
from os import path
import sys
import numpy as np
import pandas as pd
import random
import math
```

```
if not sys.platform.startswith("win"):
    raise Exception('You are on Linux! In order to run this
model you need to manually instal opendssdirect package.')

SW_VERS = hil.get_sw_version()

#path for the Python library
sendto_dir                                                =
path.expandvars(r"%APPDATA%\typhoon\{}\python_portables\p
ython3_portable\Lib\site-packages".format(SW_VERS))
add_to_python_path(os.path.normpath(sendto_dir))

#import OpenDSS library
import opendssdirect as dss
dss.utils.run_command("clear")

#path for the OpenDSS model
path = SETTINGS_DIR +"\European_MV.dss"

#compile dss model
dss.utils.run_command("Compile \"{}\"".format(path))

#initialization parameters from OpenDSS
enable = 0
# NOTE: The code specified in this handler will be executed
on simulation start.
# NOTE: Variables specified here will be available in other
handlers.
# HIL API is imported as 'hil'

global powers1
global powers2
global powers3
global powers4
global powers5
global powers6
global powers7
global powers8
global powers9
global busV1
global busV2
global busV3
global busV4
global busV5
global busV6
global busV7
global busV8
global busV9
global Q_PV1
global Totalloss
global Activeloss
global Reactiveloss
# declaring initial values to the global variable
powers1=0
powers2=0
powers3=0
powers4=0
powers5=0
powers6=0
```

```
powers7=0
powers8=0
powers9=0
Activeloss=0
Reactiveloss=0
activepower=0
reactivepower=0

phases = ["A", "B", "C"]
Q_Power=["Q_PV"]
dss.Solution.SolveSnap()
# Assign voltages of buses where PV are installed to show
in SCADA panel
dss.Circuit.SetActiveBus("bus3")
busV1 = dss.Bus.VMagAngle()
dss.Circuit.SetActiveBus("bus4")
busV2 = dss.Bus.VMagAngle()
dss.Circuit.SetActiveBus("bus5")
busV3 = dss.Bus.VMagAngle()
dss.Circuit.SetActiveBus("bus6")
busV4 = dss.Bus.VMagAngle()
dss.Circuit.SetActiveBus("bus8")
busV5 = dss.Bus.VMagAngle()
dss.Circuit.SetActiveBus("bus9")
busV6 = dss.Bus.VMagAngle()
dss.Circuit.SetActiveBus("bus10")
busV8 = dss.Bus.VMagAngle()
dss.Circuit.SetActiveBus("bus11")
busV9 = dss.Bus.VMagAngle()
dss.Circuit.SetActiveBus("bus7")
busV7 = dss.Bus.VMagAngle()

# NOTE: The code specified in this handler will be executed
on timer event.
# HIL API is imported as 'hil'

x = bool(hil.read_digital_signal(name = "enabled"))
if x:
    # Sets new active and reactive power levels in OpenDSS
for all PVsystems in network
    powers1 = hil.read_analog_signal("PVcontrol.Q")
    powers2 = hil.read_analog_signal("PVcontrol1.Q")
    powers3 = hil.read_analog_signal("PVcontrol2.Q")
    powers4 = hil.read_analog_signal("PVcontrol3.Q")
    powers5 = hil.read_analog_signal("PVcontrol4.Q")
    powers6 = hil.read_analog_signal("PVcontrol5.Q")
    powers7 = hil.read_analog_signal("PVcontrol6.Q")
    powers8 = hil.read_analog_signal("PVcontrol7.Q")
    powers9 = hil.read_analog_signal("PVcontrol8.Q")

    #Edit the reactive powers of PVs in OpenDSS model in
real time
    dss.PVsystems.Name("PV1")
    dss.PVsystems.kvar(powers1)
    #dss.PVsystems.kvar(Q_PV1)
    dss.PVsystems.Name("PV2")
    dss.PVsystems.kvar(powers2)
    dss.PVsystems.Name("PV3")
    dss.PVsystems.kvar(powers3)
```

```
    dss.PVsystems.Name("PV4")
    dss.PVsystems.kvar(powers4)
    dss.PVsystems.Name("PV5")
    dss.PVsystems.kvar(powers5)
    dss.PVsystems.Name("PV6")
    dss.PVsystems.kvar(powers6)
    dss.PVsystems.Name("PV7")
    dss.PVsystems.kvar(powers7)
    dss.PVsystems.Name("PV8")
    dss.PVsystems.kvar(powers8)
    dss.PVsystems.Name("PV9")
    dss.PVsystems.kvar(powers9)

    #Solves a snapshot of the powerflow
    dss.Solution.SolveSnap()

    if dss.Solution.Converged():
        # Selects the bus where the inverter is installed
and collect  voltages
        Totalloss=dss.Circuit.Losses()
        Activeloss=Totalloss[0]
        Reactiveloss=Totalloss[1]

        #dss.Circuit.SetActiveElement("Vsource.SOURCE")
        totalpower = dss.Circuit.TotalPower()
        activepower = totalpower[0]
        reactivepower = totalpower[1]

        # Assign voltages of buses where PV are installed
to show in SCADA panel
        dss.Circuit.SetActiveBus("bus3")
        busV1 = dss.Bus.VMagAngle()
        dss.Circuit.SetActiveBus("bus4")
        busV2 = dss.Bus.VMagAngle()
        dss.Circuit.SetActiveBus("bus5")
        busV3 = dss.Bus.VMagAngle()
        dss.Circuit.SetActiveBus("bus6")
        busV4 = dss.Bus.VMagAngle()
        dss.Circuit.SetActiveBus("bus8")
        busV5 = dss.Bus.VMagAngle()
        dss.Circuit.SetActiveBus("bus9")
        busV6 = dss.Bus.VMagAngle()
        dss.Circuit.SetActiveBus("bus10")
        busV8 = dss.Bus.VMagAngle()
        dss.Circuit.SetActiveBus("bus11")
        busV9 = dss.Bus.VMagAngle()
        dss.Circuit.SetActiveBus("bus7")
        busV7 = dss.Bus.VMagAngle()
```

2.1.1.2 **Development of panel to execute real-time control**

To run the co-simulation of Typhoon HIL and OpenDSS in real-time, real-time signals must be given from the SCADA to the OpenDSS. For this purpose, different panels with widgets are designed. Fig. 5 shows the widget setting for the slider to change the reactive power of PV in real-time. The signal processing component available in the Typhoon HIL library is used to process the signal between the cyber layer and the physical layer,

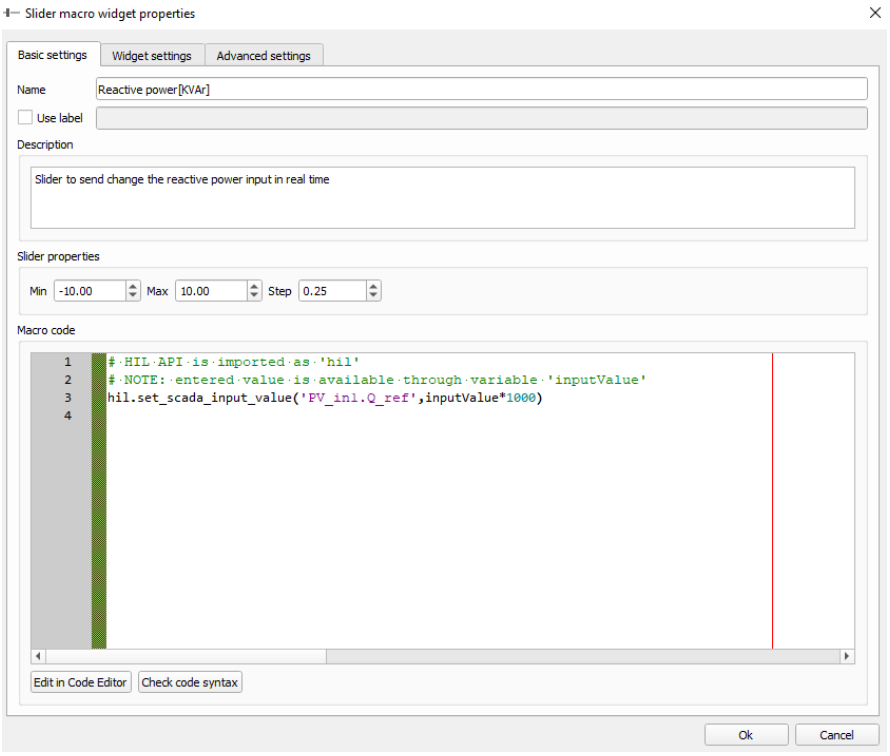

Fig. 5 Widget for changing reactive power of PV in real-time.

2.1.1.3 **Development of panel to execute real-time control**

To monitor the output variables in SCADA in real-time, a panel needs to be designed. An appropriate widget to show the desired signal is selected from the library. In the widget, a program is written to gather an appropriate signal to be monitored. Fig. 6 shows the setting of the widget to monitor the reactive power injection from SCADA in real-time. The number of this widget in SCADA depends on the number of real-time controllable PVs in the network. Similarly, Fig. 7 shows the program written in the widget to monitor the voltage at the bus where PV is connected. The number of this widget depends on how many bus voltages we need to monitor.

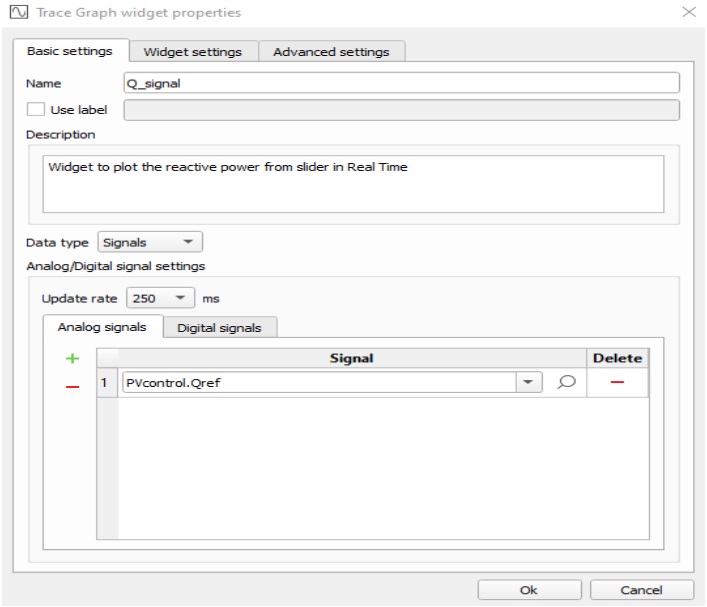

Fig. 6 Widget to display reactive power of PV in real-time.

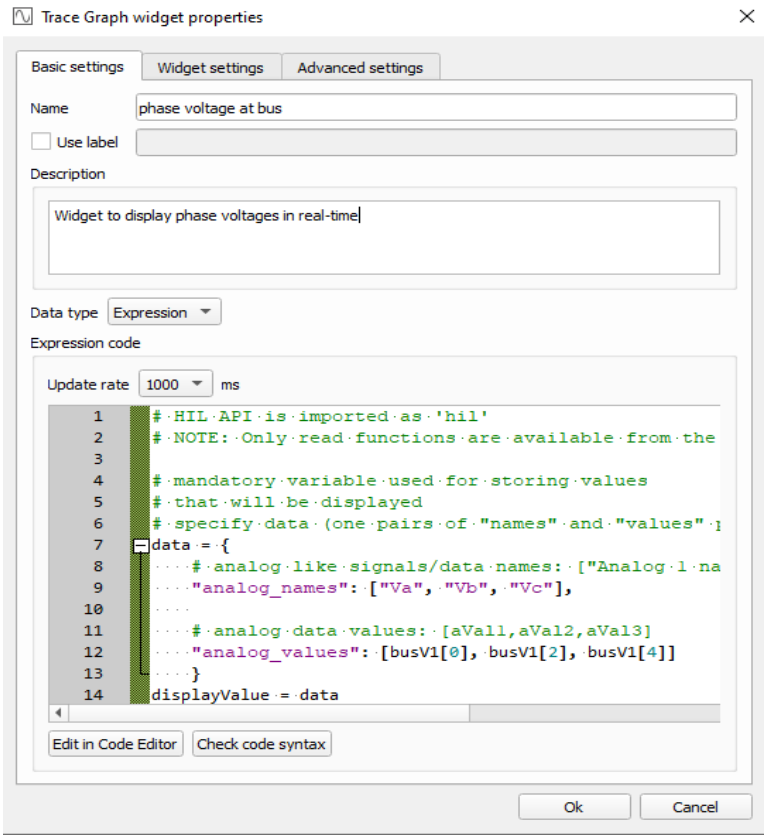

Fig. 7 Widget to display the voltage of bus in real time.

3 Implementation of Typhoon HIL and OpenDSS co-simulation framework

The overall process of implementing the co-simulation between Typhoon HIL and OpenDSS is shown in **Fig. 8** First, the test distribution system is modelled in OpenDSS. The PVs are placed on the distribution network in the OpenDSS model. The OpenDSS modules are executed through a Python interface. A Python program is written inside the SCADA of the Typhoon HIL to interact with the OpenDSS. In a schematic editor on Typhoon HIL, a communication interface between the SCADA and the Typhoon HIL real-time simulator is modelled. This model in the schematic editor can interact with the SCADA and the Typhoon HIL Real-Time Simulator. The SCADA of Typhoon HIL consists of a python program to get the signals from OpenDSS, process the signal, and display the real-time outputs. The SCADA also consists of different sliders for sending the real-time signal to the OpenDSS. At each change, the signal is fed to the OpenDSS, the load flow is executed inside the OpenDSS, and the outputs are fed into the Typhoon HIL.

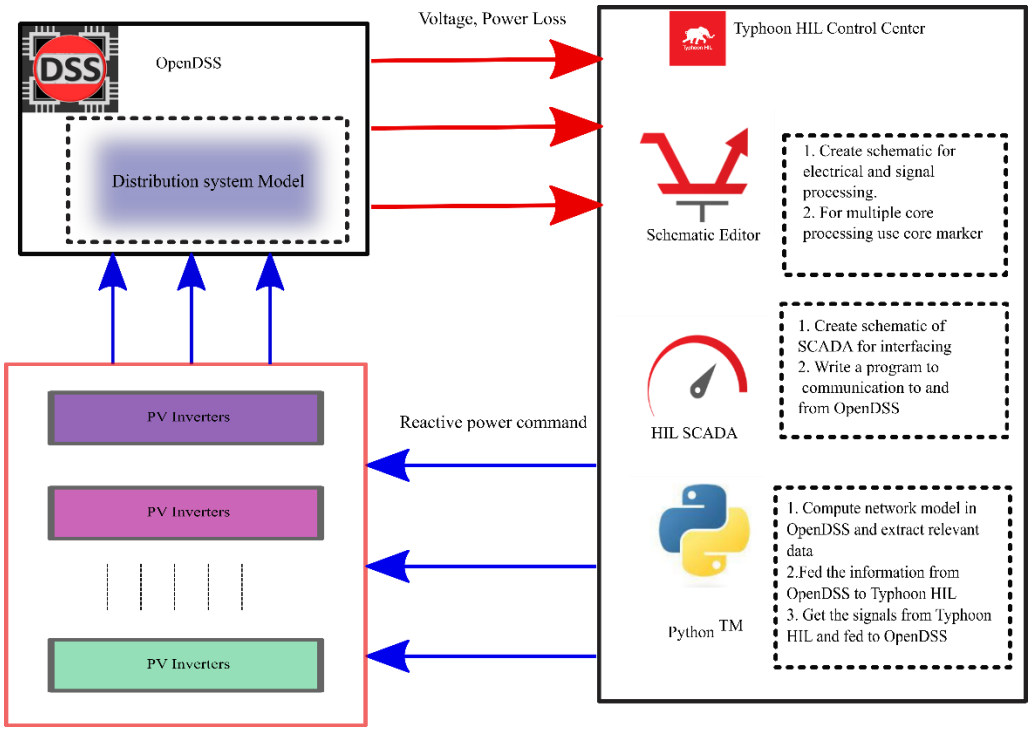

Fig. 8 Overall frameworks for co-simulation between Typhoon HIL and OpenDSS.

4 Simulation Results

The research was carried out in the Digital Energy Systems Laboratory (DIgEnSys-Lab). The DIgEnSys-Lab has physical equipment for real-time monitoring and control (see https://fglongattlab.fglongatt.org for further information). Each part of the simulation study is described in the following subsection. Typhoon HIL 604 is used to model the cybernetic and physical layers in this paper. the European medium voltage distribution network produced by CIGRE Task Force C6.04 in their publication "Benchmark Systems for Network Integration of Renewable and Distribution Energy Resources." It is assumed that the network is symmetrical and balanced. As illustrated in **Fig. 9**, the test system comprises two typical 20kV, 50 Hz, three-phase feeders named feeder 1 and feeder 2. By turning on or off the switches S1, S2, and S3, the feeder can be operated in a radial or meshed topology. In this analysis, all the switches are assumed to be closed.

The wind source considered in the original study is replaced with a PV of the same size to test the effectiveness of real-time reactive power control with smart inverters of PVs in this situation. The load and other network information are kept the same as in the original study. **Table 1** shows the ratings of PVs considered in this study.

Table 1 MV distribution network benchmark application: parameters of PV units[10].

Bus number	Type of DER	$P_{max}(kW)$
3	PV	20
4	PV	20
5	PV	30
6	PV	30
7	PV	1500
8	PV	30
9	PV	30
10	PV	40
11	PV	10

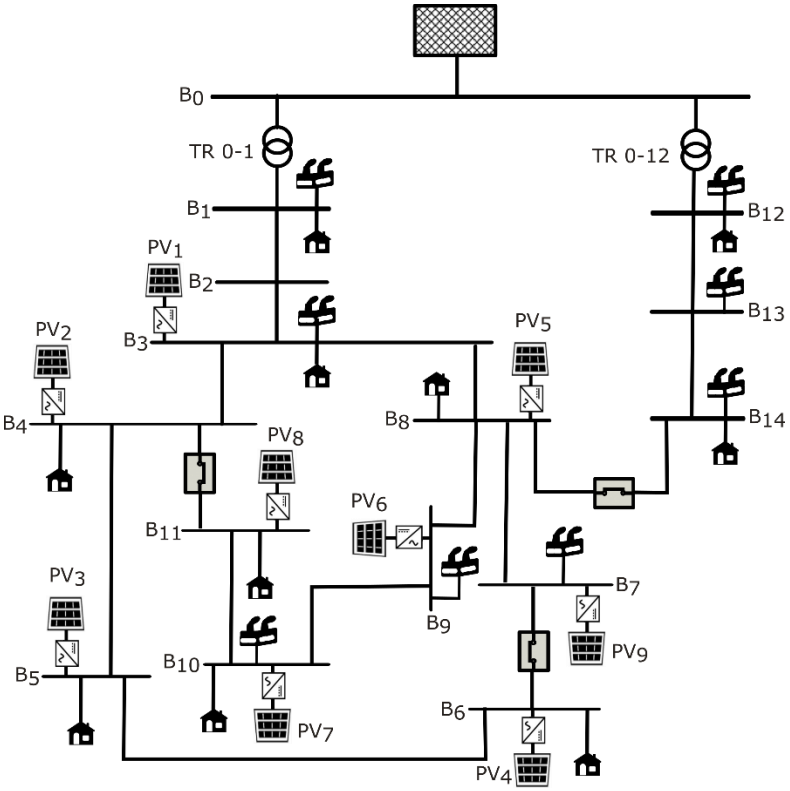

Fig. 9 Test system: Modified CIGRE medium voltage distribution system [10].

The screenshot of the SCADA of the proposed cyber-physical co-simulation framework is shown in Fig. 10. The SCADA consists of sliders to change the reactive power in real-time. The bus voltages in the network are continuously monitored during the real-time simulation. Digital and graphical displays can be inserted into the system as per the requirement of the observation.

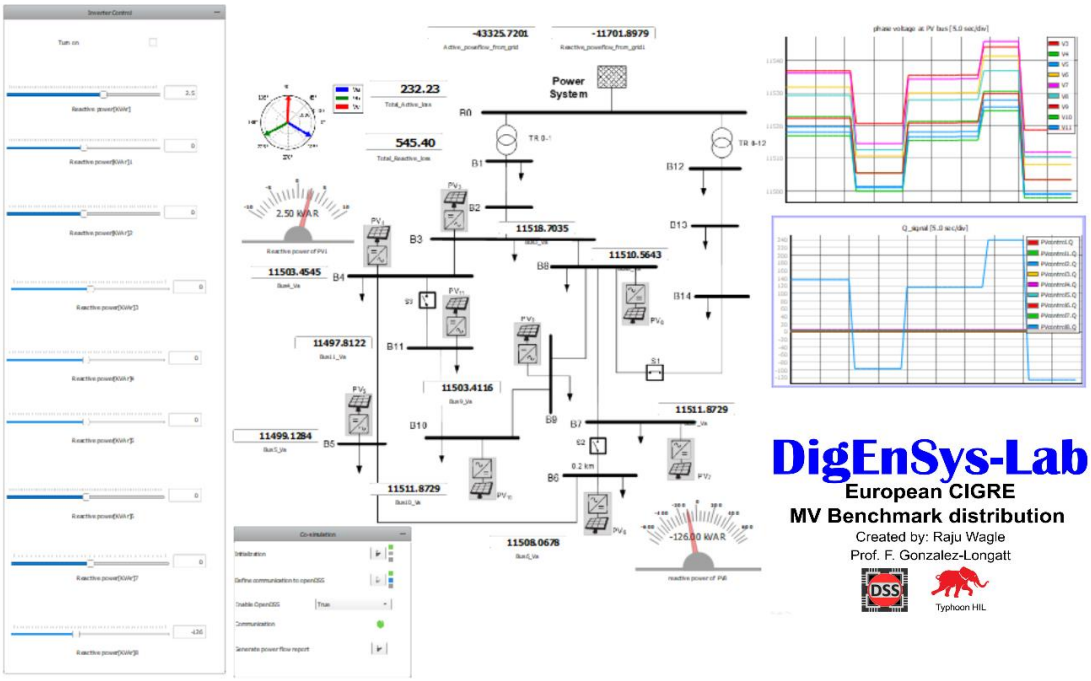

Fig. 10 Output window of proposed Cyber-physical co-simulation framework.

To demonstrate the successful operation of the proposed cyber-physical co-simulation framework, Fig. 11 shows the reactive power profile of PVs that are changed in real-time from the SCADA. In this case, the observation is made with only a change in reactive power from the PV. However, a similar observation can also be made for other controllable variables. For the change of the reactive power of the PV, Fig. 12 shows the corresponding voltage profile of the network. The voltage profile of the network changes in real-time for the real-time change in the reactive power.

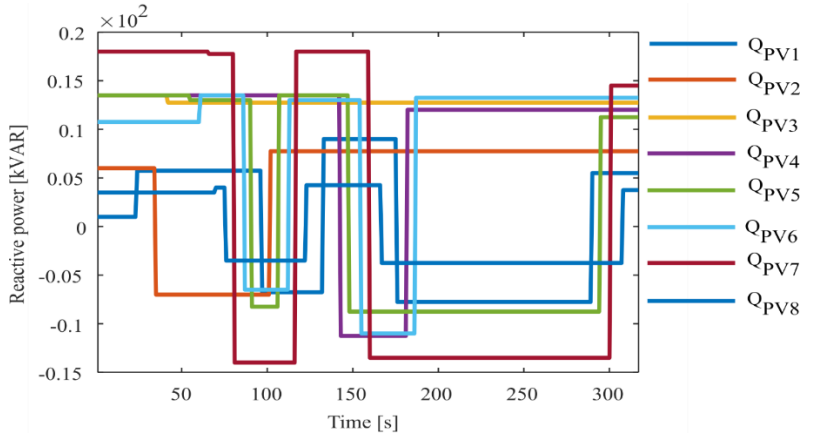

Fig. 11 Reactive power profile of PV obtained by dynamically changing the reactive power input.

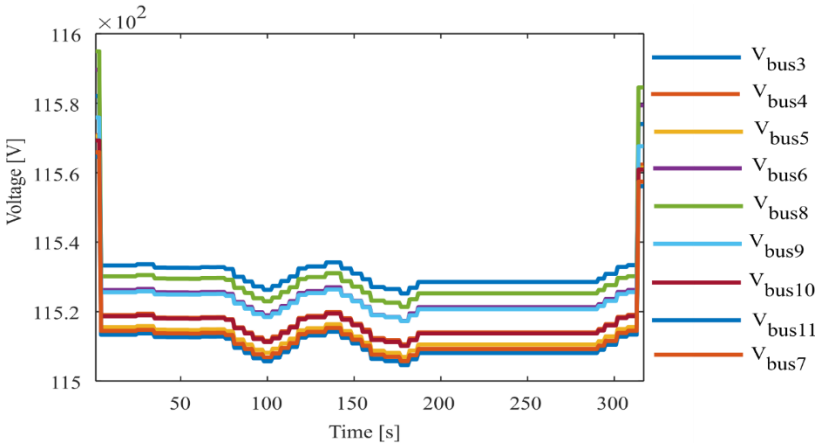

Fig. 12 Voltage profile of CIGRE network obtained from Co-simulation framework.

5 Discussion and Conclusion

This chapter proposed and demonstrated a cyber-physical co-simulation framework between Typhoon HIL and OpenDSS.

It can be easily customized and enhanced by the reader; as a consequence, it opens the door for new research approaches on real-time control and monitoring studies for distribution networks.

This research can be expanded to include real-time optimization of the distribution network with numerous PVs to determine the appropriate reactive power requirements for smart inverters and to compensate for voltage fluctuations caused by changes in loads and PV generation.

6 Open Access and Open Choice

If possible the authors would like to publish fully open access chapter.

Acknowledgments. The authors are very grateful to the Arctic Centre for Sustainable Energy (ARC) (project number 740108), UiT The Arctic University of Norway, Norway, for providing an opportunity for Mr Raju to visit and work in DIgEnSys-Lab. Authors and especially Prof F. Gonzalez-Longatt, acknowledge the technical support provided by the teams of Typhoon HIL and EPRI.

References (in Basic)

[1] F. Schloegl, S. Rohjans, S. Lehnhoff, J. Velasquez, C. Steinbrink, and P. Palensky, "Towards a classification scheme for co-simulation approaches in energy systems," *Proc. - 2015 Int. Symp. Smart Electr. Distrib. Syst. Technol. EDST 2015*, pp. 516–521, 2015.

[2] M. N. Acosta, F. Gonzalez-Longatt, M. A. Andrade, and J. R. Torres, "Optimal Reactive Power Control of Smart Inverters: Vestfold and Telemark Regional Network," in *2021 IEEE Madrid PowerTech*, 2021, pp. 1–6.

[3] C. Steinbrink, F. Schlogl, D. Babazadeh, S. Lehnhoff, S. Rohjans, and A. Narayan, "Future perspectives of co-simulation in the smart grid domain," *2018 IEEE Int. Energy Conf. ENERGYCON 2018*, pp. 1–6, 2018.

[4] A. A. van der Meer *et al.*, "Cyber-physical energy systems modeling, test specification, and co-simulation based testing," in *2017 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, 2017, pp. 1–9.

[5] V. Venkataramanan, A. Srivastava, and A. Hahn, "Real-time co-simulation testbed for microgrid cyber-physical analysis," in *2016 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, 2016, pp. 1–6.

[6] G. Cao *et al.*, "Real-time cyber−physical system co-simulation testbed for microgrids control," *IET Cyber-Physical Syst. Theory Appl.*, vol. 4, no. 1, pp. 38–45, 2019.

[7] Y. Tang *et al.*, "A hardware-in-the-loop based co-simulation platform of cyber-physical power systems for wide area protection applications," *Appl. Sci.*, vol. 7, no. 12, 2017.

[8] R. Wagle, G. Tricarico, P. Sharma, C. Sharma, J. L. Reuda, and F. Gonzalez-Longatt, "Cyber-Physical Co-Simulation Testbed for Real-Time Reactive Power Control in Smart Distribution Network," *IEEE ISGT Asia (accepted Pap.*, 2022.

[9] C. C. Sun, J. Hong, and C. C. Liu, "A co-simulation environment for integrated cyber and power systems," *2015 IEEE Int. Conf. Smart Grid Commun. SmartGridComm 2015*, pp. 133–138, 2016.

[10] Cigre, "Benchmark Systems for Network Integration of Renewable and Distributed Energy Resources," 2014.

[11] W. Sunderman, R. C. Dugan, and J. Smith, "Open source modeling of advanced inverter functions for solar photovoltaic installations," in *2014 IEEE PES T&D Conference and Exposition*, 2014, pp. 1–5.

[12] Dheepak Krishnamurthy, "OpenDSSDirect.py," *dss-extensions.org*, 2017. [Online]. Available: https://dss-extensions.org/OpenDSSDirect.py/notebooks/Example-OpenDSSDirect.py.html.

[13] T. Krechel, F. Sanchez, F. Gonzalez-Longatt, H. R. Chamorro, and J. L. Rueda, "A Transmission System Friendly Micro-grid: Optimising Active Power Losses," in *2019 IEEE Milan PowerTech*, 2019, pp. 1–6.

[14] T. HIL, "Typhoon HIL manual," *typhoon-hil*, 2022. [Online]. Available: https://www.typhoon-hil.com/documentation/.