Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

Simplifying multi-energy system co-simulations using ENERGYSIM

Digvijay Gusain [*], Miloš Cvetković, Peter Palensky

Intelligent Electrical Power Grids, TU Delft, The Netherlands

A R T I C L E   I N F O

Article history:
Received 3 September 2020
Received in revised form 8 December 2021
Accepted 17 February 2022

Keywords:
Multi energy systems
Co-simulation
Technical assessment
Hybrid simulation

A B S T R A C T

The traditional methodology for conducting technical assessments of multi-energy systems involved using domain-specific modeling tools to focus on the energy sector of interest, while making simplifying assumptions about any coupled energy sector. This was acceptable since the interactions between energy domains were minimal. However, with the expectation of an increased adoption of energy conversion technologies (such as power to X (P2X) systems: power to heat, power to gas, etc.) in the future, and consequently higher interaction between various energy sectors and stronger dependence on one another, there is a need to update the current method for conducting technical assessments. This means taking into account not only the energy sector of interest, but also any dependent energy sectors, and the associated energy transformers (P2X). In this paper, we propose a co-simulation based approach to conduct simulation-based technical assessments of multi-energy systems, which allows us to couple domain specific modeling tools. We re-introduce the tool ENERGYSIM to conduct the multi-energy system co-simulations. We motivate the importance of the proposed tool and compare it with other available tools. We highlight its main functionalities, and using a study case, we show how a multi-stakeholder, multi-energy system co-simulation can be set up and assessed.

Current code version

Current code version	2.1.7
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2020_70
Legal Code License	GNU GPL v3
Code versioning system used	git
Software code languages, tools, and services used	Python 3.6+
Operating environments tested on	Windows, Linux.
Dependencies	FMPy(0.2.14), Pandapower, PyPSA, Numpy, Pandas, Matplotlib, NetworkX, Tqdm, PyTables
Link to documentation	http://energysim.readthedocs.io
Support email for questions	d.gusain@tudelft.nl

1. Motivation and significance

The coupling of energy systems is an important driver towards a sustainable energy system. This recent trend in the coupling of various energy sectors has been driven primarily by the effort to decarbonize the energy system [1,2]. As an added benefit, the energy conversion devices enabling this coupling such as power to heat, power to gas, power to mobility, etc. can also serve as sources of flexibility to the highly renewable future power systems [3,4]. With increased interconnections, the interactions between various energy sectors are strongly coupled as well. We also refer to these coupled systems as multi-energy systems or MES.

To correctly assess MES, a holistic analysis of the system is required. It has been well noted in [5] that before an economic analysis can be conducted to determine a business case for an MES, it is essential to evaluate its technical feasibility. This involves the evaluation of various control strategies, assessment of operation and reliability of components and associated networks, determination of operational bottlenecks, etc. A modeling and simulation-based analysis, therefore, forms the first step towards the assessment of such integrated energy systems.

* Corresponding author.
E-mail address: d.gusain@tudelft.nl (Digvijay Gusain).

https://doi.org/10.1016/j.softx.2022.101021

Nomenclature

MES	Multi Energy Systems
FMI	Functional Mockup Interface
FMU	Functional Mockup Unit
CTDS	Continuous Time Dynamic Simulator
EN	Electrical Network
HN	Heat Network
GN	Gas Network
CHP	Combined Heat and Power
EV	Electric Vehicles
P2H	Power to Heat
P2G	Power to Gas
WTO	Wind Turbine Operator

Simulation-based assessments are not new in energy system analysis. The dynamic and steady-state characteristics of different energy domains are unique. For example, the system dynamics for different energy carriers evolve at different time scales. For an electricity network, the changes in active power, and hence frequency, are immediately visible throughout the network. For a heat network, this is not the case: while pressure changes in the heat network are reflected throughout the network in seconds, the temperature dynamics across the network can take minutes and hours to reach a steady state. Consequently, to accurately consider these characteristics in analyzing a particular energy sector, state-of-the-art tools and solvers have been used. These tools have been developed using years of research and development to be able to accurately model the aforementioned unique characteristics of the energy domain in focus. Examples of such proprietary tools include PowerFactory for modeling electrical power system, Dymola/OpenModelica for modeling fluid system, MATLAB and Python-based APMonitor [6] for developing model-based control and PyTorch, [7] for developing data-driven controls.

When it comes to conducting a combined system study for an MES, there are two natural pathways. The first pathway is modeling the entire system in a single modeling environment, such as MATLAB, and then simulating it using a general-purpose solver. This is a time-consuming approach: it requires extensive knowledge on the part of the modeler to design different components of the MES. Because the MES is modeled in a single environment, it will be solved using a single, general-purpose solver. Such a solver may not be well suited to solving unique dynamics of subsystems within the MES, and therefore provide results that are not as accurate as with models and solvers developed using domain specific modeling tools. Additionally, simulating a large and complex multi domain model can require significant computing resources [8].

The second pathway is by dividing the MES into smaller subsystems (such as electrical subsystem, heat subsystem, gas subsystem, etc.), and leveraging software techniques to couple these subsystems. The subsystems can be modeled using domain-specific modeling tools, and solved using dedicated solvers (instead of general-purpose solvers) to obtain high accuracy results. This method is known as co-simulation, or coupled simulation [9]. Co-simulation is a method that allows the coupling of models developed in various modeling environments by managing the time progression of the simulation and coordinating the data flow between subsystem models. By exchanging data (values of interest), such as process outputs, sensor measurements, etc. between subsystem models, dynamic interaction between the subsystems can be facilitated. Even though any data value can be shared, when creating an MES model, the exchanged variables are usually those which lay on the boundary of two energy carriers. For example mechanical power of steam turbine obtained from thermo-mechanical model (process output) of generator given as input to the rotor of the synchronous generator in the electrical power system model, or temperature of room obtained from a building thermal model (sensor measurement) given as input to a control system model, etc. Enabling this dynamic interaction allows for a holistic analysis of the system, where domain-specific characteristics are preserved while interacting with other energy domains. To reduce computational burden, techniques such as parallel and distributed computing can also be used [10] to speed up a co-simulation. As is noted in [11,12], operational model details have significant impact on results. Thus, by using domain-specific accurate models, potentially misleading simplifying assumptions are removed. Although, it must be noted that splitting larger systems into smaller subsystems and using co-simulation introduces some numerical challenges of its own, such as algebraic loops [9].

The main challenge in achieving this goal of a holistic system assessment is the development of a modular framework that allows the coupling of various subsystem models and an algorithm that manages overall simulation time progression and data exchange. This is where energysim steps in. energysim allows users to easily couple subsystem models and focus on high-level tasks in MES technical assessment studies such as subsystem model development, control algorithm development, case study definition, etc. rather than focusing on co-simulation specific tasks such as management of time progression in the co-simulation and data exchange between subsystem models. The availability of a simplified energy system co-simulator will allow increased insights into an MES setting by enabling a more collaborative modeling and simulation environment.

Previously, we used energysim in [13,14]. In [13], we demonstrated how complex the model of the closed-cycle gas turbine was combined with the dynamic model of the electricity network for analyzing the impact of fuel supply change on electrical network frequency. In [14] we demonstrated how energysim can facilitate a multi-stakeholder analysis that involved concurrent simulation of detailed models of the electricity grid, electrolyzer, and control systems to correct forecasting errors of a nearby wind turbine. The version of energysim used for simulating use cases in the aforementioned articles has been majorly revamped. The current version (energysim (v2.1.7)) provides more simulation adapters (explained in the next subsection) to couple other widely used energy modeling tools, allows access to the algorithm that coordinates time progression and data exchange which allows users to implement non-energy applications, uses HDF data format to store results, making it useful for very-high fidelity simulations generating considerable amounts of data.

energysim is developed in Python and is compatible with any version above v3.6 and can be easily installed with the python package manager (pypi). A use case has been provided in the main repository and a working example of the use case described in this paper has also been uploaded to the Code Ocean platform and is available as supplementary material.

Comparison with other tools

At the heart of any co-simulation tool lies an algorithm that manages time progression and data exchange. There already exist a few tools in literature such as Mosaik [15], MasterSim [16], MESCOS [17], Ptolemy II [18] to set up co-simulation. However, we believe, energysim offers several advantages to its users when compared to existing aforementioned tools. The first and foremost advantage is that it is developed in Python. Python is the most widely used language for scientific and general-purpose computing and has developed a huge user base, especially in the energy system community. Proprietary tools also frequently provide bindings to Python, which makes accessing them easy. This familiarity with the programming language allows ENERGYSIM to be easily understandable and accessible to large audiences compared to tools which are developed in languages such as JAVA (PTOLEMY II), C, C++ (MASTERSIM, MESCOS). Secondly, the structure of ENERGYSIM is modular. This means that access to time progression and message exchange algorithm can be done via what we refer to as *simulation adapters*. Simulation adapters interface the simulation entity to ENERGYSIM by defining four key function: `init()`, `set_value()`, `get_value()`, and `step()`. These functions enable ENERGYSIM to initialize the simulator, set and get variable values at any time, and control simulation progress of the simulator in time respectively. To put it simply, these adapters provide a way for ENERGYSIM to "talk to simulators". The tool already provides ready-made adapters to couple the most common tools used in the energy system modeling and simulation domain, however, creating new adapters can also be done easily. This is in contrast to other tools which require complex setup configurations (for ex. setting up Scenario API and Component API for each simulator with MOSAIK) or can support only a single type of simulation entities (for ex. MASTERSIM supports only Functional Mock-up Units (FMUs)).

Main contribution

The main contribution of this work is the proposal of a new and improved co-simulation framework, ENERGYSIM, that specifically addresses the needs of conducting technical simulations of complex MES.

2. Software description

2.1. Software architecture

ENERGYSIM is classified as a "hybrid-simulator". It supports both quasi-static and continuous-time dynamic simulation (CTDS). We have not classified ENERGYSIM as a discrete-time simulator because the term encompasses a broad range of simulation techniques, some of which ENERGYSIM does not support, for example, event-based simulations, such as those involving communication network simulation.

2.1.1. Time progression and data exchange management

In ENERGYSIM, there are two main time variables to define: macro-time step (for overall c-simulation) and micro-time step (for individual simulators). The data exchange between simulators occurs at fixed time intervals, known as the macro-time step. Between each macro-time step, individual simulators use an optional and unique micro-time step for solving their own model equations. This is essential for CTDS models to perform time integration for solving their model equations. The flowchart describing the co-simulation is shown in Fig. 1.

In between the macro-time steps, when no input data is available to the CTDS model, an interpolation method needs to be applied. Although there are quite a few techniques to implement the interpolation (constant, linear, polynomial [19]), within ENERGYSIM, we have opted for a constant interpolation method. In this method, the inputs to the CTDS model are held constant at the value obtained at the last macro-time step. We are currently also testing new functionality that will allow users to select between different interpolations.

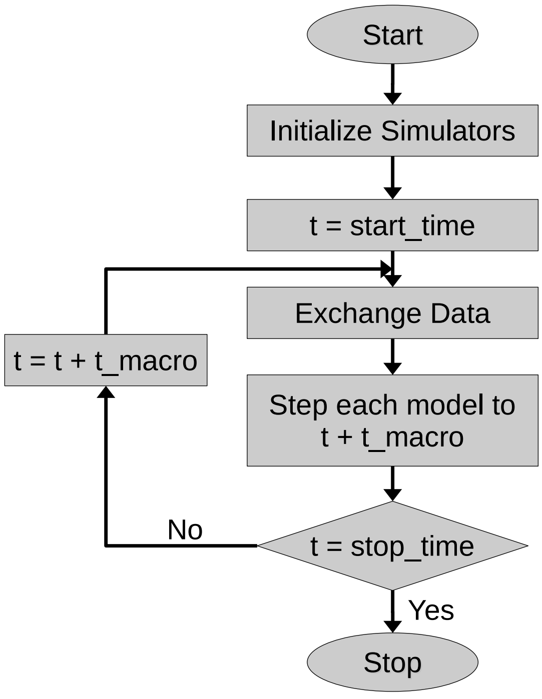

Fig. 1. Flowchart depicting the co-simulation process. To start, the simulators are initialized and then, at each macro time step data is exchanged between the simulators. In between the macro-time steps, each simulator can use an individual micro-time step for simulation.

2.1.2. Software components

The core component of the ENERGYSIM package is the `world` object. Once the `world` object is imported from ENERGYSIM package, the user can instantiate it as shown in listing 1.

Listing 1: Instantiating world

```
from energysim import world
my_world = world(start_time=0, stop_time=23*3600, logging=
    True, t_macro=60)
```

`my_world` is the canvas on which simulators, simulator connections, and simulation options can be specified. The main parameters to be specified here are `start_time`, `stop_time`, `logging`, and `t_macro`. The parameter `t_macro` specifies the macro-time step and has a default value of 60 s.

Once `my_world` object is created, users can add simulators to it via the `add_simulator()` method as shown in listing 2.

Listing 2: Adding simulators

```
my_world.add_simulator(sim_type = 'fmu', sim_name = 'fmu1',
      sim_loc = '/path/to/fmu', step_size = 1, inputs = [],
      outputs = ['var', 'obj1.var1'])
```

The `add_simulator()` method requires specification of `sim_type`, `sim_name`, `sim_loc`, and `step_size`. The parameter `step_size` specifies the micro-time step, unique to each simulator. These six parameters are shared for the specification of any simulator and are enough to execute a basic co-simulation. However, if additional parameters need to be specified, users can also do so. For example, a power flow model of an electric network (modeled using PANDAPOWER [20]) added in ENERGYSIM using `sim_type = 'powerflow'`, by default, uses

Table 1
List of common modeling languages used in energy system modeling and simulation community.

Energy domain	Modeling language
Heat	Modelica, MATLAB, Simulink, Python, CSV, Others
Electricity	Modelica, Python, PowerFactory, Simulink, Others
Gas	Python, Modelica, Simulink, Others
Transportation	Python, MATLAB, CSV, Others

the AC power flow functionality of PANDAPOWER to solve the model (ENERGYSIM does not contain a solver of its own). Therefore, an additional argument `pf` can be specified while adding the simulator so that if required, users can specify a different power flow options. For PANDAPOWER models, these include "pf" (default), "dcpf", "opf", and "dcopf". This is useful when, for example, an optimal controller based on electrical network power flow is needed. The example in the GitHub repository uses this functionality. There exist other similar simulator-specific options within ENERGYSIM. A detailed list and description of these options are provided in the software documentation.

As was already mentioned in Section 1, to couple simulators to ENERGYSIM, we have developed simulation adapters. Through an extensive literature search, we identified the languages that are most commonly used to develop models for energy domains of interest: Modelica [21–25], MATLAB [5], Python [26,27] are typically used to model thermal and heat systems. Python [28,29], DigSILENT PowerFactory [22,30,31], Modelica [32,33], and MATLAB [34] are used for electricity network simulations. For gas systems, Modelica [32,33,35] is a common choice of software for many, whereas, for analysis of electric vehicle systems, Modelica [36], Python [26,36,37], and MATLAB [38] are used the most. These are summarized in Table 1.

Some modeling languages allow exporting models as functional mock-up units (FMUs) according to the FMI standard [39]. Models are packaged as a combination of XML files, binaries, and C-code and distributed as a ZIP file called FMU. The FMU contains encrypted model equations and optionally an associated solver. Model exchange via FMU is gaining wider adoption across the modeling and simulation community, with currently more than 150 tools supporting exporting models to FMUs. Within ENERGYSIM, FMUs can be added by setting the value of the `sim_type` parameter as `'fmu'`.

To ensure wider operability and integration to other software/python packages, we have also provided a way for users to interface their own simulator of choice with ENERGYSIM by setting the `sim_type` parameter with `'external'`. To add a user-defined simulator to ENERGYSIM via user-developed simulation adapter, the to-be-coupled simulator must offer a "play-and-pause" functionality. This means that ENERGYSIM must be able to:

- initialize the simulator,
- get inputs from simulator when requested,
- instruct the simulator to step forward in time,
- request output values from the simulator,
- pause the simulation while the simulator waits for instructions, and new inputs from ENERGYSIM.

A detailed description of this method is available in the documentation.

Once the simulators are specified and added to `my_world`, users can specify the connections between the simulators as a

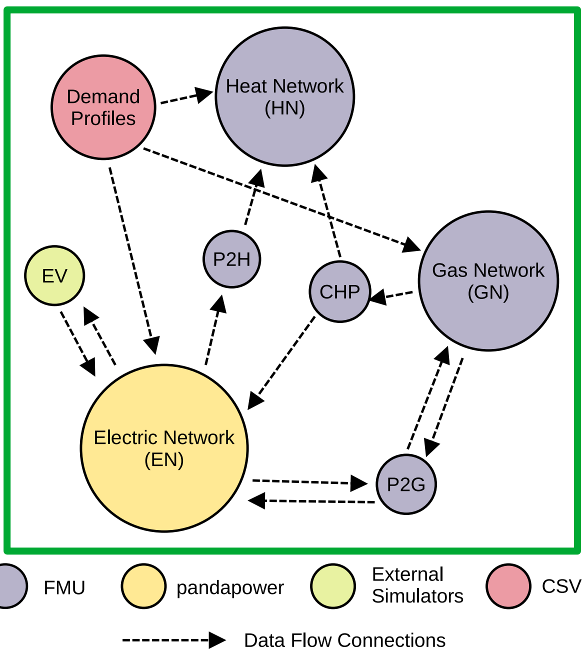

Fig. 2. An example of an MES model and dummy interactions between subsystem models setup using ENERGYSIM. Variables can be load active power, thermal power demand, gas demand, sent from Demand Profile simulator to EN, HN, and GN simulators. The EV simulator can exchange active power, reactive power, voltage value, charging status, etc. with the EN. The P2G simulator can exchange active power set-points with EN and gas flow rates and pressure with GN.

python dictionary object, as shown. Then, the `my_world` object can then be simulated using the `simulate()` command. By default, the `record_all` parameter is set to *True* which instructs ENERGYSIM to record each simulator's output values at every micro-time step. *False* leads to the recording of output only at macro-time step intervals. The results can be obtained by calling the command `my_world.results()`, which returns a Python dictionary object with keys as `sim_name` and value as a pandas dataframe with time-stamped output values. If the `to_csv` option is set to *True*, then results are also exported as csv files. Parameter `pbar` toggles the simulation progress bar.

Listing 3: Finalizing simulation

```
connections = {'sim1.output_variable1' : 'sim2.
     input_variable1', 'sim3.output_variable2' : 'sim4.
     input_variable2', 'sim1.output_variable3' : 'sim2.
     input_variable3',}
my_world.add_connections(connections)
my_world.simulate(pbar=True, record_all=True)
results = my_world.results(to_csv=False)
```

Fig. 2 provides a graphical overview of an example of MES configured using ENERGYSIM.

2.2. Software functionalities

Apart from the basic functionality to add simulators and simulate them, ENERGYSIM offers a range of additional inbuilt functions to support the user in setting up multi-energy co-simulations.

2.2.1. Adding signals

In many cases, users may require a simple external input to their model. For example, a control system simulator needs a constant input value of *True* to indicate that the simulator is active, or a wind power plant simulator which may require a random number generated from a continuously updated probability distribution function as an input to account for the randomness of wind power production, etc. These inputs can be added as CSV simulators, FMU simulators, or first, be modeled in another modeling language and interfaced to ENERGYSIM using the simulation adapters. However, such a process can be cumbersome. To simplify this process, we use "signals". Signals can provide the users with requested values and can be defined as python functions. Signals can be added to ENERGYSIM using the `add_signal()` method. Examples of both time-dependent and independent signals are shown in listing 4.

Listing 4: Adding signal

```
def td_signal(t):
    return [True]

def tid_signal(t):
    return [np.sin(2*np.pi*omega*t)]

my_world.add_signal(sim_name='sine_wave_signal', signal =
     td_signal, step_size=1)
my_world.add_signal(sim_name='constant_signal', signal =
     tid_signal, step_size=1)
```

2.2.2. Initialization

For CTDS models, correct initialization of system states is needed for accurate assessments. Different initial values will result in different dynamic behavior. To specify initial values to simulators, users can provide the initial values of parameters for the simulators as a dictionary with `'init'` keyword. This dictionary can then be passed onto the `my_world` object through `options` method. Due to limited space, we have not included a listing for the same, but it can be found in the documentation.

2.2.3. Modifying data exchange

On many occasions, it becomes necessary to modify the output value of a particular simulator before it is provided to the other simulator. This situation is fairly common in multi-energy system simulations. Consider two simulators: a thermo-mechanical model of a combined heat and power system (CHP) and a steady-state power flow model of an electric network (EN). The power output from the CHP simulator is obtained in Watts (W). This power output of the CHP needs to be provided to the corresponding generator model in the EN. However, the generator in the EN accepts values only in megawatts (MW). One way to address the problem is to change the output values of the CHP in the model itself and recompile the model. However, this may not always be possible to do so (model may be encrypted). Therefore, ENERGYSIM provides a method to apply linear modifications given by Eq. (1).

$$y^* = a \cdot y + b \tag{1}$$

where y is the output of the simulator to be modified and y^* is the modified output. By default, b is 0, therefore, users can provide a list with either just $[a]$ or both $[a, b]$. A dictionary entry to the `options` dictionary can be provided to ENERGYSIM using the `'modify_signal'` keyword. Both initialization and output modification is shown in listing 5.

Listing 5: Initialization and Output Modifications

```
ini = {'sim1':(['sim_vars'], [vals]),
       'sim2':(['sim_vars'], [vals])}

mdf = {'sim1.var1':[a],
       'sim2.var1':[a, b]}

options = {'init' : ini,
           'modify_signal': mdf}

my_world.options(options)
```

2.2.4. Parameter sweep

Understanding the sensitivity of results on various simulation variables is an important part of technical assessments of MES. To perform repeated simulations with different initial conditions for the co-simulation in focus, users can make use of `'init'` option shown in Section 2.2.2. By using a for-loop and providing different values of initial conditions, the sensitivity of results from the co-simulation can be obtained. A more sophisticated functionality is in the pipeline for the next release.

2.2.5. Topology plot

Using `plot()` method on `my_world` object, users can obtain a graph plot of the simulators, and connections between them. ENERGYSIM uses python's NetworkX package to generate the topological plot of the multi-energy system based on the specified connections' dictionary.

3. Illustrative examples

3.1. Multi-stakeholder analysis

Consider an 18MW rated wind turbine connected to an electricity distribution grid. To mitigate its wind power fluctuations throughout the day, the wind turbine operator, WTO, forms a bilateral contract with a nearby 10MW industrial hydrogen electrolyzer to absorb its fluctuations. The electrolyzer's main goal is to produce hydrogen for its industrial process, and therefore, it can only modify its 10MW power set-point by ±5MW. Furthermore, the electrical grid operator needs to make sure that any sudden changes in load and generation setpoints for the electrolyzer and the wind turbine do not affect voltage stability in the distribution grid. We, therefore, need to conduct a technical analysis on the ability of the electrolyzer to adequately provide flexibility, while ensuring that electrolyzer production constraints are met, and the grid stability is not endangered. A central control system is designed which continuously receives information from various entities. These include: wind power forecast ($P_{w,fc}$) from WTO (added as `csv` simulator), hydrogen pressure and mass flow rate ($p_{H2}, \dot{m}_{H2}$) from electrolyzer (added as a `fmu` simulator), and actual wind power production of wind turbine ($P_{w,a}$) and grid bus voltages (V) from pandapower network (added as `powerflow` simulator). Based on the received values, it calculates the electrolyzer power set point ($P_{el,sp}$) and dispatches it to the electrolyzer and grid. This setpoint is used by 1) the electrical grid to evaluate bus voltage based on the AC power flow solution, and 2) by electrolyzer to calculate hydrogen production rates. In the current setup, the controller also has an additional input for emergency control (u_e) (added as a `signal` simulator) to stop the electrolyzer operation in any case of an emergency. The system setup is shown in Fig. 3.

Listing 6 highlights the code to set up the co-simulation.

Listing 6: Multi-stakeholder analysis

```
from energysim import world

mw = world(stop_time=3600*5., logging = True, t_macro =
     120)

simulators_dir = ''

controller_loc = os.path.join(simulators_dir, '
     controller_continuous.fmu')
grid_loc = os.path.join(simulators_dir, 'gridModel.p')
electrolyser_loc = os.path.join(simulators_dir, '
     Electrolyser_Subsystem.fmu')

mw.add_simulator(sim_type = 'powerflow', sim_name = 'grid1'
     , sim_loc = grid_loc, inputs = ['wind12.P', '
     Electrolyser.P'], outputs=['Bus 0.V', 'Bus 1.V', 'Bus
     12.V', 'wind1.P', 'wind12.P', 'Electrolyser.P'],
     step_size=3)
mw.add_simulator(sim_type = 'fmu', sim_name = 'controller',
      sim_loc = controller_loc, step_size = 3, inputs = ['v
     ', 'P', 'E_c'], outputs=['y', 'gain1.y', '
     load_should_be', 'new_load_should_be', 'p_forecasted',
      'power_delta'])
mw.add_simulator(sim_type = 'fmu', sim_name = 'electrolyser
     ', sim_loc = electrolyser_loc, step_size = 3, inputs =
      ['p'], outputs=['y1', 'y2', 'p', 'integrator1.y'],
     variable=True)
mw.add_simulator(sim_type = 'csv', sim_name = 'wind_data',
     sim_loc= os.path.join(simulators_dir, 'diff_win.csv'),
      step_size=900, outputs=['speed', 'power', 'power2'])

def emergency(time):
    return [1]

mw.add_signal(sim_name = 'Emergency', signal = emergency)

options = {'init':{'controller':(['v', 'E_c', 'P', 'C_max'
     ], [14, 1, -18, -100]),
                   'electrolyser':(['p'], [10])},
            }

mw.options(options)

connections = {'wind_data.speed': 'controller.v',
               'wind_data.power': 'grid1.wind12.P',
               'wind_data.power2': 'grid1.wind1.P',
               'controller.y':'grid1.Electrolyser.P',
               'Emergency.y':'controller.E_c',
               'grid1.wind12.P':'controller.P',
               'grid1.Electrolyser.P':'electrolyser.p',
               }

mw.add_connections(connections)

res = mw.simulate(pbar=True)
```

The simulation takes exactly 6 min to execute on a dual-core Intel i5-6300U CPU @ 2.4 GHz running Ubuntu 20.04. Figs. 4–6 shows the electrical network bus voltage, the hydrogen production rate obtained from the detailed electrolyzer model simulator, and the electrolyzer operating power set-points determined by the control system simulator respectively.

4. Impact

The need to take the proposed approach of coupled simulation towards energy system technical analyses is highlighted by the recent blackouts in Texas. Although the impact of extreme weather on power systems was assessed, the impact on natural gas supply and its equipment, which acts as a fuel source for power generation, was not considered simultaneously. This

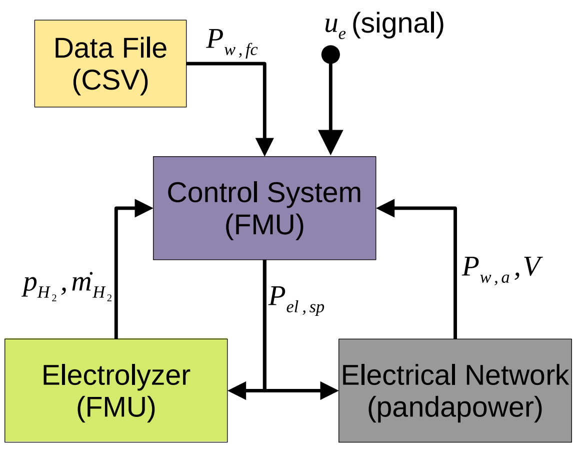

Fig. 3. The co-simulation setup for Section 3.1. The Data Files provide information regarding forecasted wind power to the control system. The Electrolyzer FMU provides information such as internal pressure and mass flow rates to the control system. The electrical network provides the voltage values at each bus node and the actual power output from the wind-powered plant. The controller uses this information to determine the operational power setpoint for the electrolyzer which it conveys to both the electrical power grid and the electrolyzer model.

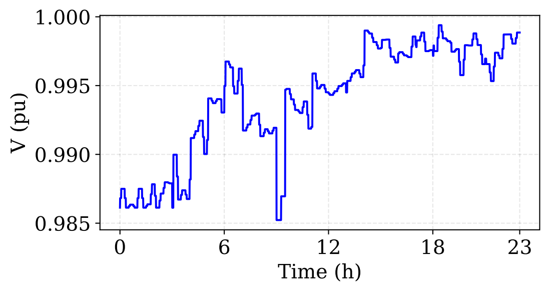

Fig. 4. Voltage at the bus where electrolyzer is connected to the electrical grid. This output comes from the electrical network simulator.

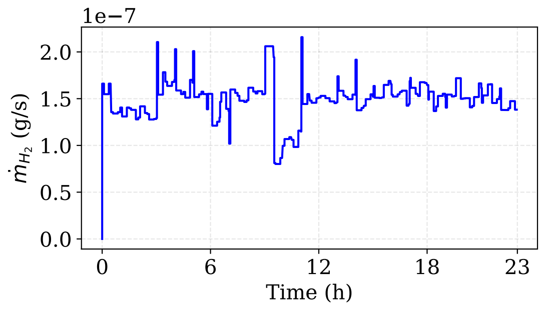

Fig. 5. Production of hydrogen gas from the variable operation of electrolyzer. This output comes from the detailed electrolyzer model simulator.

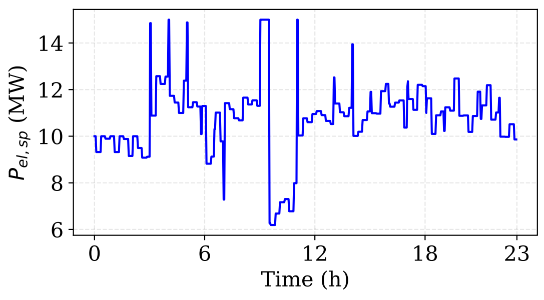

Fig. 6. Electrolyzer operating power set point obtained. This output comes from control system simulator.

led to inaccurate assessments, leading to a catastrophic situation whereby millions of people lost access to electricity and heat. [40].

Using ENERGYSIM, a combined assessment of energy sectors is possible, allowing users to get a better understanding of technical limitations in the interconnected system. The existence of such a tool facilitates interdisciplinary research efforts by bringing together models developed by experts in modeling tools of their choice. By making the technical assessment model agnostic, all subsystems can be modeled in desired detail and coupled with dedicated solvers to obtain a closer to reality system, and consequently accurate solutions. This is in contrast to existing domain-specific modeling and simulation tools, which either greatly simplify or completely ignore the interactions and dependence of energy sectors.

Using this tool, we expect the users to couple models of components and systems in different energy domains to derive critical insights and knowledge in designing and operating an integrated energy system. The ease of use will allow users to focus on tasks such as identification of operational bottlenecks, testing control strategies, the sensitivity of various factors on system behavior, checking technical feasibility of operating P2X devices on both the electricity and coupled sector simultaneously, etc.

Currently, ENERGYSIM has enabled students and researchers at the Delft University of Technology to couple different types of energy systems for various studies. These include studying the impact of model fidelity on the availability of flexibility from P2X resources, development of machine learning models of integrated energy systems, and the evaluation of predictive control for flexibility coordination in multi-energy system setup. There has also been an interest in using ENERGYSIM as a model for testing reinforcement learning-based control strategies.

Outside the intended user-group of energy system modeler and analysts, we can also expect ENERGYSIM to be used in other applications requiring coupling of subsystem models to determine the evolution of system behavior over time. This could be enabled by using `external` simulators to access the time progression and data management algorithm of ENERGYSIM.

5. Future work

ENERGYSIM is a versatile tool for researchers to integrate various energy subsystem models to conduct holistic and comprehensive technical assessments. However, despite a range of available functionality, a few functionalities are still lacking, which are being actively worked on. These tasks have been identified to increase the adoption of ENERGYSIM as well as to increase its capabilities as a multi-energy system co-simulator. Among these include the development of a method to conduct parameter sensitivity, the ability to specify different interpolations for data exchange between simulators, a method to parallelize simulation of individual simulators to increase computational efficiency and enable large-scale co-simulations, development of simulation adapters for other common software tools such as DigSILENT PowerFactory. Additionally, we intend to make available more case-study and examples as the ENERGYSIM user group increases.

6. Conclusions

In this paper, we presented ENERGYSIM, a multi-energy system co-simulation tool for coupling energy system models developed in different software tools. We have described in detail the main features and functionalities of the proposed software, illustrating its ease of use and versatility. We provided code listings to highlight the important features of the proposed tool: such as adding user-defined signals, initialization for conducting parameter sweeps, ability to perform output data modification, and generating topology plots for complex and simple MES. In the end, we demonstrated the use of ENERGYSIM in a case study where detailed models of an electrolyzer, the electricity network, and a control system are used to assess the ability of the electrolyzer to act as a flexibility service provider to the wind plant generator. With the provided case study, it was shown how various stakeholders can come together to perform a holistic study and use preferred modeling tools and solvers in doing so, something which may not be possible with traditional monolithic simulation tools.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

We would like to thank NWO for funding the project Heat and Power Systems at Industrial Sites and Harbors (HaPSISH).

References

[1] Müller C, Falke T, Hoffrichter A, Wyrwoll L, Schmitt C, Trageser M, et al. Integrated planning and evaluation of multi-modal energy systems for decarbonization of Germany. Energy Procedia 2019;158:3482–7, URL https://www.sciencedirect.com/science/article/pii/S1876610219309671.

[2] Burandt T, Xiong B, Löffler K, Oei P-Y. Decarbonizing China's energy system – Modeling the transformation of the electricity, transportation, heat, and industrial sectors. Appl Energy 2019;255:113820, URL https://www.sciencedirect.com/science/article/pii/S0306261919315077.

[3] Dall'Anese E, Mancarella P, Monti A. Unlocking flexibility: Integrated optimization and control of multienergy systems. IEEE Power Energy Mag 2017;15(1):43–52. http://dx.doi.org/10.1109/MPE.2016.2625218, Conference Name: IEEE Power and Energy Magazine.

[4] Arabzadeh V, Mikkola J, Jasiunas J, Lund PD. Deep decarbonization of urban energy systems through renewable energy and sector-coupling flexibility strategies. J Environ Manag 2020;260:110090. http://dx.doi.org/10.1016/j.jenvman.2020.110090, URL https://www.sciencedirect.com/science/article/pii/S0301479720300281.

[5] Ommen T, Markussen WB, Elmegaard B. Comparison of linear, mixed integer and non-linear programming methods in energy system dispatch modelling. Energy 2014;74:109–18. http://dx.doi.org/10.1016/j.energy.2014.04.023, URL http://www.sciencedirect.com/science/article/pii/S0360544214004368.

[6] APMonitor Optimization Suite URL http://apmonitor.com/.

[7] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An imperative style, high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A, dAlché Buc F, Fox E, Garnett R, editors. Advances in neural information processing systems, vol. 32. Curran Associates, Inc.; 2019, URL https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

[8] Novák P, Wimmer M, Kadera P. Slicing simulation models into co-simulations. In: Mařík V, Wahlster W, Strasser T, Kadera P, editors. Industrial applications of holonic and multi-agent systems. Lecture notes in computer science, Cham: Springer International Publishing; 2017, p. 111–24. http://dx.doi.org/10.1007/978-3-319-64635-0_9.

[9] Palensky P, Van Der Meer AA, Lopez CD, Joseph A, Pan K. Cosimulation of intelligent power systems: Fundamentals, software architecture, numerics, and coupling. IEEE Ind Electr Mag 2017;11(1):34–50. http://dx.doi.org/10.1109/MIE.2016.2639825, Conference Name: IEEE Industrial Electronics Magazine.

[10] Kraft J, Meyer T, Schweizer B. Reduction of the computation time of large multibody systems with co-simulation methods. In: Schweizer B, editor. IUTAM Symposium on solver-coupling and co-simulation. IUTAM Bookseries, Cham: Springer International Publishing; 2019, p. 131–52. http://dx.doi.org/10.1007/978-3-030-14883-6_8.

[11] Morales-España G, Ramírez-Elizondo L, Hobbs BF. Hidden power system inflexibilities imposed by traditional unit commitment formulations. Appl Energy 2017;191:223–38. http://dx.doi.org/10.1016/j.apenergy.2017.01.089, URL http://www.sciencedirect.com/science/article/pii/S0306261917301009.

[12] Helistö N, Kiviluoma J, Morales-España G, O'Dwyer C. Impact of operational details and temporal representations on investment planning in energy systems dominated by wind and solar. Appl Energy 2021;290:116712. http://dx.doi.org/10.1016/j.apenergy.2021.116712, URL https://www.sciencedirect.com/science/article/pii/S0306261921002312.

[13] Gusain D, Cvetković M, Palensky P. Energy flexibility analysis using fmuworld. In: 2019 IEEE milan powertech. 2019, p. 1–6. http://dx.doi.org/10.1109/PTC.2019.8810433.

[14] Gusain D, Cvetković M, Bentvelsen R, Palensky P. Technical assessment of large scale PEM electrolyzers as flexibility service providers. In: 2020 IEEE 29th international symposium on industrial electronics. 2020, p. 1074–8. http://dx.doi.org/10.1109/ISIE45063.2020.9152462, ISSN: 2163-5145.

[15] Schütte S, Scherfke S, Tröschel M. Mosaik: A framework for modular simulation of active components in smart grids. In: 2011 IEEE first international workshop on smart grid modeling and simulation. 2011, p. 55–60. http://dx.doi.org/10.1109/SGMS.2011.6089027.

[16] Evora Gomez J, Hernández Cabrera JJ, Tavella J-P. Semantic interoperability in co-simulation: use cases and requirements. In: 30th European simulation and modelling conference , ESM 2016, octubre, las palmas de gran canaria, spain, p. 5-9. 2016, URL https://accedacris.ulpgc.es/jspui/handle/10553/56248.

[17] Molitor C, Groß S, Zeitz J, Monti A. MESCOS—A multienergy system cosimulator for city district energy systems. IEEE Trans Ind Inf 2014;10(4):2247–56. http://dx.doi.org/10.1109/TII.2014.2334058, Conference Name: IEEE Transactions on Industrial Informatics.

[18] Wetter M. Co-simulation of building energy and control systems with the building controls virtual test bed. J Build Perform Simul 2011;4(3):185–203. http://dx.doi.org/10.1080/19401493.2010.518631, Publisher: Taylor & Francis.

[19] Schweizer B, Lu D, Li P. Co-simulation method for solver coupling with algebraic constraints incorporating relaxation techniques. Multibody Syst Dyn 2016;36(1):1–36. http://dx.doi.org/10.1007/s11044-015-9464-9, URL http://link.springer.com/10.1007/s11044-015-9464-9.

[20] Thurner L, Scheidler A, Schäfer F, Menke J, Dollichon J, Meier F, et al. Pandapower — An open-source python tool for convenient modeling, analysis, and optimization of electric power systems. IEEE Trans Power Syst 2018;33(6):6510–21. http://dx.doi.org/10.1109/TPWRS.2018.2829021.

[21] Liu C, Cheng M-S, Zhao B-C, Dai Z-M. A wind power plant with thermal energy storage for improving the utilization of wind energy. Energies 2017;10(12):2126. http://dx.doi.org/10.3390/en10122126, URL https://www.mdpi.com/1996-1073/10/12/2126.

[22] Leitner B, Widl E, Gawlik W, Hofmann R. A method for technical assessment of power-to-heat use cases to couple local district heating and electrical distribution grids. Energy 2019;182:729–38. http://dx.doi.org/10.1016/j.energy.2019.06.016, URL http://www.sciencedirect.com/science/article/pii/S0360544219311399.

[23] Widl E, Leitner B, Basciotti D, Henein S, Ferhatbegovic T, Hofmann R. Combined optimal design and control of hybrid thermal-electrical distribution grids using co-simulation. Energies 2020;13(8):1945. http://dx.doi.org/10.3390/en13081945, URL https://www.mdpi.com/1996-1073/13/8/1945.

[24] Neirotti F, Noussan M, Riverso S, Manganini G. Analysis of different strategies for lowering the operation temperature in existing district heating networks. Energies 2019;12(2):1–17, URL https://ideas.repec.org/a/gam/jeners/v12y2019i2p321-d199375.html.

[25] Vialle S, Tavella J-P, Dad C, Corniglion R, Caujolle M, Reinbold V. Scaling FMI-CS based multi-simulation beyond thousand FMUs on infiniband cluster. 2017, p. 673–82. http://dx.doi.org/10.3384/ecp17132673, URL http://www.ep.liu.se/ecp/article.asp?issue=132%26article=74.

[26] Backe S, Korpås M, Tomasgard A. Heat and electric vehicle flexibility in the European power system: A case study of norwegian energy communities. Int J Electr Power Energy Syst 2021;125:106479. http://dx.doi.org/10.1016/j.ijepes.2020.106479, URL https://www.sciencedirect.com/science/article/pii/S0142061520322079.

[27] Lohmeier D, Cronbach D, Drauz SR, Braun M, Kneiske TM. Pandapipes: An open-source piping grid calculation package for multi-energy grid simulations. Sustainability 2020;12(23):9899. http://dx.doi.org/10.3390/su12239899, URL https://www.mdpi.com/2071-1050/12/23/9899.

[28] Phongtrakul T, Kongjeen Y, Bhumkittipich K. Analysis of power load flow for power distribution system based on pypsa toolbox. In: 2018 15th International conference on electrical engineering/electronics, computer, telecommunications and information technology. 2018, p. 764–7. http://dx.doi.org/10.1109/ECTICon.2018.8619954.

[29] Bogunović N, Vlahinić S, Franković D, Komen V. Application of PandaPower tool in evaluating the potential of using PV distributed generation for voltage regulation in electrical power networks. In: 2020 43rd International convention on information, communication and electronic technology. 2020, p. 912–7. http://dx.doi.org/10.23919/MIPRO48935.2020.9245182, ISSN: 2623-8764.

[30] López CD, Cvetković M, Palensky P. Distributed co-simulation for collaborative analysis of power system dynamic behavior. In: Mediterranean Conference on power generation, transmission, distribution and energy conversion. 2018, p. 1–5. http://dx.doi.org/10.1049/cp.2018.1876.

[31] López CD, van der Meer AA, Cvetković M, Palensky P. A variable-rate co-simulation environment for the dynamic analysis of multi-area power systems. In: 2017 IEEE Manchester powertech. 2017, p. 1–6. http://dx.doi.org/10.1109/PTC.2017.7981117.

[32] Aguilera M, Vanfretti L, Bogodorova T, Gómez F. Coalesced gas turbine and power system modeling and simulation using modelica. In: Proceedings of the American modelica conference 2018. Linköping University Press; 2019, URL http://dx.doi.org/10.3384/ecp1815493.

[33] Heckel J-P, Becker C. Advanced modeling of electric components in integrated energy systems with the TransiEnt library. 2019, p. 759–68. http://dx.doi.org/10.3384/ecp19157759, URL https://ep.liu.se/en/conference-article.aspx?series=ecp&issue=157&Article_No=79.

[34] Degefa MZ, Sæle H, Andresen C. Analysis of future loading scenarios in a norwegian LV network. In: 2019 International conference on smart energy systems and technologies. 2019, p. 1–6. http://dx.doi.org/10.1109/SEST.2019.8849081.

[35] Chen B, Hajimolana YS, Venkataraman V, Ni M, Aravind PV. Integration of reversible solid oxide cells with methane synthesis (ReSOC-MS) in grid stabilization: A dynamic investigation. Appl Energy 2019;250:558–67. http://dx.doi.org/10.1016/j.apenergy.2019.04.162, URL https://www.sciencedirect.com/science/article/pii/S0306261919308220.

[36] Stûber M. Simulating a variable-structure model of an electric vehicle for battery life estimation using modelica/dymola and python. 2017, p. 291–8. http://dx.doi.org/10.3384/ecp17132291, URL https://ep.liu.se/en/conference-article.aspx?series=ecp&issue=132&Article_No=31.

[37] Wulff N. Emobility in Python (emobpy) and vehicle energy consumption in Python (VencoPy). Demonstration of two open source tools describing electric vehicle energy demand. Germany (online); 2020, URL https://elib.dlr.de/135072/.

[38] Barreras JV, Fleischer C, Christensen AE, Swierczynski M, Schaltz E, Andreasen SJ, et al. An advanced HIL simulation battery model for battery management system testing. IEEE Trans Ind Appl 2016;52(6):5086–99. http://dx.doi.org/10.1109/TIA.2016.2585539, Conference Name: IEEE Transactions on Industry Applications.

[39] Blochwitz T, Otter M, Arnold M, Bausch C, Clauß C, Elmqvist H, et al. The functional mockup interface for tool independent exchange of simulation models. In: Clauß C, editor. Proceedings of the 8th international modelica conference. Dresden: Linköping University Press; 2011, p. 105–14, ISSN: 1650-3740.

[40] 2021 Texas power crisis. 2021, Page Version ID: 1044087165,