Challenges in Transitioning from Co-Simulation to Practical Application: A Case Study on Economic Emission Dispatch in a Greenhouse Compartment

Christian Skafte Beck Clausen [1[0000-0003-3118-7253]], Sebastian Lehnhoff[2[0000-0003-2340-6807]], Jan Sören Schwarz[2[0000-0003-0261-4412]], Bo Nørregaard Jørgensen [1[0000-0001-5678-6602]] and Zheng Grace Ma [1[0000-0002-9134-1032]]

[1] SDU Center for Energy Informatics, Maersk Mc-Kinney Moeller Institute, The Faculty of Engineering, University of Southern Denmark, Odense, Denmark
csbc@mmmi.sdu.dk
bnj@mmmi.sdu.dk
zma@mmmi.sdu.dk

[2] OFFIS – Institute for Information Technology, Oldenburg, Germany
sebastian.lehnhoff@offis.de
jan.soeren.schwarz@offis.de

Abstract. Co-simulation is a widely applied method used to analyze the behavior of complex, interdisciplinary, and integrated cyber-physical control systems. Despite its prevalence, the transition from co-simulated control systems into practical applications is not discussed as much in the literature. This leaves a gap in the literature because practitioners may not be aware of these challenges. This paper aims to uncover and discuss some of the challenges that arise in the transition from a co-simulated control system to a practical application.

A case study on economic emission dispatch in a Danish industrial greenhouse compartment serves as the fundament in studying these challenges. Economic emission dispatch is a method that can be used in a closed-loop arrangement to decrease costs and emissions of multiple energy production units. The case study is first implemented as a co-simulation which is subject to a subsequent practical implementation. The co-simulation implementation is governed by the open-source framework mosaik that is used extensively in smart grid applications. In contrast, the practical implementation is not governed by mosaik due to architectural design discrepancies. A key feature of the study is the use of software-in-the-loop, which means that the controller being tested is the actual software intended for deployment.

The highlighted challenges include that the core abstractions (master algorithm, scenario-script, and protocol) of the co-simulation framework cannot be transferred to an operational context due to design discrepancies. Despite these challenges, the co-simulation can still serve as a baseline for comparing functional performance metrics during the transition.

Keywords: Economic dispatch, co-simulation, software-in-the-loop, SIL, cyber-physical system, transition, challenges, greenhouse, simulation-to-reality gap

1 Introduction

The transition to sustainable energy solutions necessitates the adoption of advanced digital technologies (IEA, 2023). A prevalent method in this context is co-simulation, which integrates various simulation models to analyze the behavior of complex, interdisciplinary, and integrated cyber-physical control systems (Gomes et al., 2018; Van Der Meer et al., 2017; Vogt et al., 2018). Despite its widespread use in studying novel control strategies in simulated environments, there is limited literature exploring its transition to practical control applications (Samad et al., 2020). This forms a gap between researchers that develop novel control strategies and practitioners that implement them. This is a problem because practitioners may not be aware of the challenges and underlying assumptions of co-simulation when transferring this technology to practical applications. The general problem is that co-simulation technology is typically designed for the purpose of development, testing, and validation in a context that may not reflect operational constraints. Eventually, practitioners may encounter implementation issues that remain hidden until late in development such as integration problems, system interoperability (protocols), concurrency, synchronization, performance and resource constraints. The goal and contribution of this paper is to investigate and discuss these challenges through a case study. A key feature of the study is the use of software-in-the-loop, which means that the controller being tested is the actual software intended for deployment. The study aims to assist researchers in transitioning their co-simulations to practical applications. The motivation is that there is a need to develop a testing approach that mirrors real-world design conditions, because existing approaches are limited in this aspect (Alleyne et al., 2023; Van Der Meer et al., 2017). To achieve this, two implementations of a case study on closed-loop economic emission dispatch in a Danish greenhouse compartment are compared by their design and results. The co-simulation implementation is governed by the co-simulation framework mosaik (Ofenloch et al., 2022; Steinbrink et al., 2019), while the practical implementation is oriented towards operational constraints and is therefore not governed by mosaik. Section 1.1 introduce the background and related work for undertaking the study. Section 2 describes the research method. Section 3 showcases the case study with the local energy system of an industrial greenhouse compartment. Section 4 details the design and implementation. Section 5 presents the results of the study. Section 6 discusses the identified challenges and limitations between the two implementations and Section 7 concludes the study and provides suggestions for future work.

1.1 Background and related work

This paper extends the previous work of economic emission dispatch (EED) within an industrial greenhouse compartment (Clausen et al., 2024a). EED is an approach used to determine the optimal output of multiple generation units to meet the demand at the lowest possible cost and emissions, subject to operational constraints (Chowdhury & Rahman, 1990; Hassan et al., 2022; Mahdi et al., 2018; Rizk-Allah et al., 2023). In the previous work, the EED was defined as a multi-objective optimization problem, and more concretely, implemented as a Multi-Objective Genetic Algorithm (MOGA)

(Coello et al., 2007). The multi-objective optimization extended EED by considering additional objectives, such as minimizing CO2 and meeting the energy demands within the greenhouse compartment. It was shown that this approach can provide more flexible management of the trade-offs between different business goals, providing a set of Pareto-optimal solutions from which decision-makers can select the most suitable option based on their priorities. However, the previous work did not couple the optimized EED outputs to the greenhouse's local energy system to form a cyber-physical control system. This paper advances the realization of such as system.

A cyber-physical system (CPS) consists of collaborating computational entities connected with the Internet, the physical world and its on-going processes (Monostori, 2014). A closed-loop CPS observes and controls physical processes in a continuous loop to maintain predictable and stable operation of the system. A system that controls energy resources is classified as a CPS because software and physical processes are undeniably coupled. In EED, the closed-loop control maintains optimal operation by dynamically adjusting energy resource dispatch in response to operational changes like energy demands, markets, and forecasts. This adaptive capability is essential for managing the variability and uncertainty inherent in many modern energy systems like EED for greenhouses.

Testing an EED application for greenhouses is non-trivial because of the inherent dynamics like weather, energy demands, seasonal variation, economics, technological integration, yield impact, and CO2 emissions. These conditions make it unrealistic to develop such system in the field without preliminary testing methods. Simulation-based methods like co-simulation are suitable for this task.

Co-simulation is an approach that utilize diverse simulation models by connecting them to test their emergent behavior in a scenario prior to field-testing. In the energy domain, co-simulation has been used to test the performance by simulating the complex energy systems such as smart grids (Vogt et al., 2018). At early stages, a co-simulation can serve to demonstrate a conceptual design using coarse simulation components. At this stage, the control component is typically implemented within the modeling environment, referred to as Model-in-the-loop (MIL). However, a co-simulation can be gradually refined by increasing its fidelity, which is the degree to which it mirrors the real system, by exchanging models with more detailed ones. Substituting the control component with the actual control software implementation is called Software-in-the-loop (SIL) (Clausen et al., 2024b). SIL testing is focused on the software implementation to ensure that the software behaves correctly when interacting with other components. Further in-the-loop extensions include processor-in-the-loop (PIL), power-hardware-in-the-loop (PHIL) and hardware-in-the-loop (HIL), all which gradually moves the simulation closer to real-world operating conditions (Clausen et al., 2024b; Maniatopoulos et al., 2017). The literature offers limited research on the progressive refinement process, evolving from early-stage co-simulation to field-tested application. For this paper, SIL is sufficient because this is the first step towards the realization of EED.

2 Research Method

This study combines EED, co-simulation, and SIL testing approaches to advance the realization of a closed-loop CPS for optimal energy dispatch in greenhouses. A case study forms the basis for designing and implementing two different architectures with identical functionality – a co-simulation implementation and a practical implementation. The co-simulation implementation is used to deduct a functional baseline to verify and compare the practical implementation. The purpose of the practical implementation is to reflect the operational design constraints at a higher fidelity through SIL testing. If the output from each architecture is similar, confidence in the practical architecture increases. The research approach is illustrated in Fig. 1.

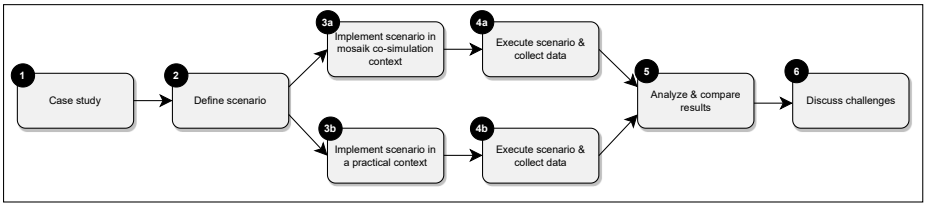

Fig. 1. Illustration of the research method.

Both architectures have overlapping technological constraints. This is the case for the simulation models and the optimization framework. The simulation models are reused from a previous study of a greenhouse project (Yang, 2022). These simulation models were developed in Dymola/Modelica and subsequently exported as Functional Mock-up Units (FMUs) using the Functional Mock-up Interface (FMI) model exchange format. (Blochwitz et al., 2012). The FMI is an open standard, that ensures compatibility and interoperability across tools, and therefore enables the models to be reused across various simulation environments. An FMU is a package that implements the FMI standard, and it encapsulates its dynamic model and, in some cases, the solver. The optimization framework used within the controller was developed at the SDU Center for Energy Informatics and employs a multi-objective optimization algorithm (Clausen & Sørensen, 2021). Similar multi-objective optimization frameworks are available, such as jMetal and MOEA Framework, but they lack certain features i.e., (i) a decision-making module that can select ideal solutions from the Pareto front and (ii) integration with FMI/FMU. These features were demonstrated in a previous study (Clausen et al., 2024a).

Moving on to the technological discrepancies between the two architectures. The co-simulation implementation is governed by the mosaik co-simulation framework (Steinbrink et al., 2019). Mosaik is a renowned open-source co-simulation framework in Python, that has been used extensively in analyzing smart grid applications (Ofenloch et al., 2022; Steinbrink et al., 2019). Mosaik solves hard design problems like configuration, deployment, dependencies, concurrency, and interoperability including synchronization, protocol, and interfacing. Furthermore, it features a user-friendly Application Programming Interface (API) and can integrate simulators implemented in various programming languages and has FMI/FMU support. Lastly, mosaik

supports co-simulation of closed-loop control. The HELICS co-simulation framework was also considered but is more complex to set up, with unnecessary extra features for this research (Hardy et al., 2024).

The practical implementation uses the NATS.io open-source middleware to achieve low-latency real-time interoperability between components (Sharvari & Sowmya, 2019). NATS.io exchanges data segmented in messages and features multimodal sync/async communication paradigms like streams, event-based, request-based, distributed key/value pairs, and microservices. Similar communication middleware could be used instead like MQTT, gRPC, RSocket, and AMQP. However, NATS.io's support for multiple communication paradigms is appealing because it meets a broad range of communication needs without requiring the use of different technologies. Furthermore, NATS.io is part of the Cloud Native Computing Foundation (CNCF) ensuring community support, vendor neutrality, and a high degree of interoperability with other systems.

3 Case Study: Economic Emission Dispatch in an Industrial Greenhouse Compartment

The case study involves EED in an industrial greenhouse compartment located in Denmark. The local energy system of the greenhouse compartment is illustrated in Fig. 2. The local energy system consists of multiple energy sources: A gas boiler (GB), a heat pump (HP), district heating (DH), a combined heat- and power plant (CHP), and the distribution grid. The produced heat is stored in a thermal energy storage (TES) which enables strategic utilization because the TES can be used as a buffer in scenarios of fluctuating energy prices and CO2 emissions. The compartment is connected to the distribution grid from where it consumes all its electrical power. The power produced by the CHP is sold to the distribution grid due to national regulations. Each of the energy sources can be controlled individually through setpoint control variables.

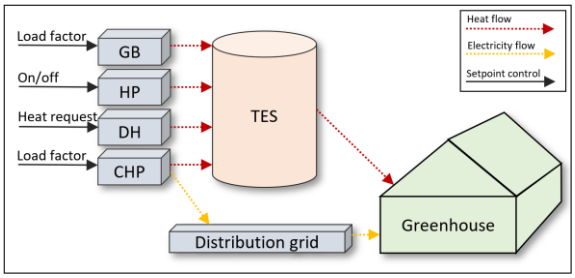

Fig. 2. The local energy system of the greenhouse compartment.

3.1 High-level View of the Cyber-Physical Control System

A controller is assigned the responsibility of computing the optimal setpoints. Fig. 3 depicts the bidirectional relationship between the controller and the local energy system. This relationship forms a closed control loop, where the controller receives the

recent state of the energy system, computes optimal setpoints, and sends the setpoints to the local energy system.

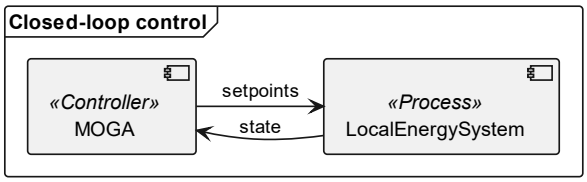

Fig. 3. A high-level view of the closed-loop CPS.

The controller incorporates an economic dispatch algorithm formulated as a Multi-Objective Problem (MOP) and implemented as a MOGA. The MOP aims at minimizing two primary objectives: (i) the total costs, and (ii) the total CO2 emissions while satisfying both the greenhouse's heat- and electricity demands, and the physical constraints of the system. The formulation of the MOP is described in Section 3.2. The MOP minimize the objectives over a future period, which requires that the algorithm has access to both forecast data and a model that can estimate the behavior of the local energy system. In other words, the forecast data and the energy system models are utilized to evaluate candidate setpoint vectors against the objectives. The closed control loop procedure is formulated in Algorithm 1. The algorithm starts with the initial state of the local energy system, and its goal is to produce a vector of setpoints and it operates within an endless control loop (line 1). Firstly, the algorithm retrieves forecasts for the greenhouse's heating and electricity demands (line 2). It also gathers forecasts for energy prices, including electricity, gas, and district heating (line 3). Additionally, it collects forecasts for CO2 emissions related to these energy sources (line 4). These forecasts along with the current state (line 5) are used to initialize the set of objectives that guide the optimization process (line 6). The optimization process starts with an empty Pareto frontier intended for containing non-dominated setpoints (line 7) and iteratively generate new candidate setpoints (line 9). The candidate setpoints are evaluated against the objectives (line 10) and the non-dominated setpoints are added to the Pareto frontier (line 11). This process is iterated until the termination criterion is reached. The termination criterion is bound to a fixed deadline of 30 minutes. This deadline can be tuned but is kept relatively large partly because of the inertia of the local energy system and to allow the MOGA to converge towards optimality. After the optimization loop concludes, a decision-making strategy is employed to select the most suitable compromise solution from the Pareto frontier (line 13). This selected compromise represents the vector of setpoints that will be used to control the local energy system (line 14). The entire process repeats, continuously adapting to new forecasts and system states to maintain optimal control of the local energy system.

ALGORITHM 1: ECONOMIC EMISSION DISPATCH CLOSED-LOOP CONTROL

Input: Initial state of the local energy system S_{init}
Output: A vector of setpoints
while *(controlling)*
  F_{heat}, $F_{electricity}$ ← retrieve greenhouse heat- and electricity demands forecasts
  $F_{price\_electricity}$, F_{price_gas}, $F_{price\_district\_heating}$ ← retrieve energy prices forecasts
  $F_{co2\_electricity}$, F_{co2_gas}, $F_{co2\_district\_heating}$ ← retrieve CO2 emissions forecasts
  state ← retrieve the current state of the local energy system
  objectives ← initializeObjectives(forecasts, state)
  pf ← empty set
  while *optimization not terminated* ***do***
    candidates ← generate candidate setpoints
    candidates' ← evaluate(*candidates*, *objectives*)
    pf ← updateParetofront(*pf*, *candidates'*)
  end while
  selectedCompromise ← decisionMaker(*pf*, *strategy*)
  return *selectedCompromise*
end while

Each objective estimates the future behavior of the local energy system by exploiting models and forecast data. The length of this future is determined by a variable-sized sliding window. The window size can vary from 12 to 35 hours which reflects the day-ahead spot market of electricity prices provided by Nord Pool (NordPoolGroup) and the day-ahead CO2 emissions provided by the Danish agency Energinet (Energinet).

3.2 Local Energy System Models

This section presents the models of the local energy system. The models are reused from previous work (Clausen et al., 2024a; Yang, 2022). The local energy system was implemented in Dymola/Modelica and exported as FMI 2.0 compatible FMUs. Fig. 4 shows a block diagram of the composed models, their relationships, and the input/output variables of each model. A description of the models input/output variables are provided in Table 1. The models were derived using grey-box modeling techniques based on on-site measurements from a commercial greenhouse in Denmark. In contrary, white-box models can provide high fidelity but are computationally intensive, and black-box modeling depend on high-quality training data, which were not available. The grey-box models approximate the system's response by tuning theoretical models with collected measurements. The system dynamics were simplified to linear equations, and the thermal energy storage does not consider temperature gradients or heat loss. These dynamics increase model fidelity at increased computational costs.

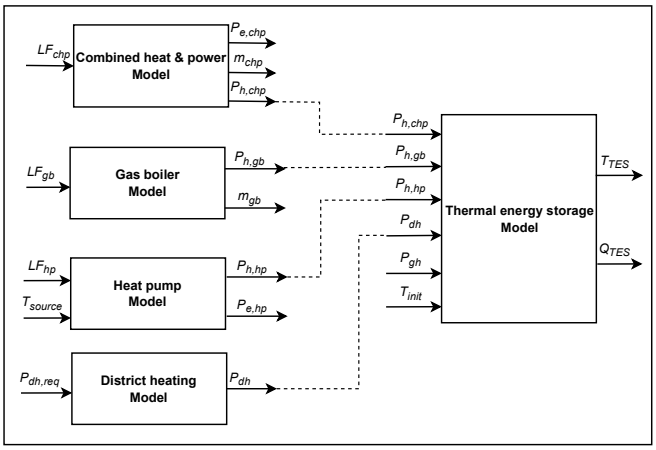

Fig. 4. Block diagram of the compartment's local energy system (Clausen et al., 2024a).

The input vector used to instantiate the energy system is defined by

$$S_{init} = [LF_{chp}, LF_{gb}, LF_{hp}, T_{source}, T_{init}, P_{dh,req}, P_{gh}] \quad (1)$$

from which the output of each model is calculated. The heat output vector P_{out} = [$P_{h,chp}$, $P_{h,gb}$, $P_{h,hp}$, P_{dh}] is used as input parameters of the TES. The outputs E_{out} = [$P_{e,chp}$, m_{chp}, m_{gb}, $P_{e,hp}$, P_{dh}] are used to calculate the economics of the energy system in terms of costs and CO_2 emissions.

Table 1. I/O structure of the FMU models.

Model	I/O	Symbol	Unit	Range	Initial value	Description
CHP	Input	LF_{chp}	[-]	[0, 1]	0.0	Part load ratio of the CHP
	Output	$P_{h,chp}$	[MW]	[0, 2.72]	0.0	Heat power produced by the CHP.
	Output	$P_{e,chp}$	[MW]	[0, 1.28]	0.0	Electric power produced by the CHP.
	Output	m_{chp}	[kg/s]	[0, 0.11]	0.0	Natural gas mass flow rate consumed by the CHP.
GB	Input	LF_{gb}	[-]	[0, 1]	0.0	Part load ratio for the gas boiler.
	Output	$P_{h,gb}$	[MW]	[0, 7]	0	Heat power produced by the GB.
	Output	m_{gb}	[kg/s]	[0, 0.194]	0.0	Natural gas mass flow rate consumed by the gas boiler.
HP	Input	LF_{hp}	[-]	0 or 1 (binary)	0	Load factor of the HP model. Off=0, On=1.
	Input	T_{source}	[°C]	[10, 30]	15	Temperature at the inlet of the evaporator of the HP.
	Output	$P_{h,hp}$	[kW]	[0, 500]	0	Heat power produced by the HP.
	Output	$P_{e,hp}$	[kW]	[0, 125]	0	Electricity consumed by the HP.
DH	Input	$P_{dh,req}$	[MW]	[0, 6]	0.0	Requested heat from the district heating network.

	Output	P_{dh}	[MW]	[0, 5.4]	0.0	Obtained heat from the district heating network (90% efficiency).
TES	Input	$P_{h,chp}$	[MW]	[0, 2.72]	0.0	Heat power produced by the CHP.
	Input	P_{dh}	[MW]	[0, 5.4]	0.0	Obtained heat from the district heating network (90% efficiency).
	Input	$P_{h,hp}$	[kW]	[0, 500]	0	Heat power produced by the HP.
	Input	$P_{h,gb}$	[MW]	[0, 7]	0	Heat power produced by the GB.
	Input	P_{gh}	[MW]	[0, 10]	0	Greenhouse heat demand.
	Input	T_{init}	[°C]	[43.96, 79.84]	50.0	The initial temperature of the thermal energy storage (TES).
	Output	T_{TES}	[°C]	[43.96, 79.84]	50.0	Water temperature of the TES.
	Output	Q_{TES}	[MWh]	[0, 64.4]	10.84	Stored energy in the TES.

3.3 Multi-Objective Problem Definition

This section presents the MOP definition of the EED optimization that are reused from (Clausen et al., 2024a). The energy dispatch schedule is represented as a decision vector that contains the energy systems load factors for each instant in the schedule. The schedule is a prescription of how to operate the facility over an extended period, and therefore, the subvector describing components at index $i = 0$ holds the setpoints that are actuated next.

$$S = [C_i, C_{i+1}, \dots, C_n, G_i, G_{i+1}, \dots, G_n, H_i, H_{i+1}, \dots, H_n, D_i, D_{i+1}, \dots, D_n]^T \quad (2)$$

where S is the schedule, i is an instant, n is the number of instants, C is the CHP load factor, G is the GB load factor, H is the HP load factor and D is the heat request to the DH provider. The number of instants n is dependent on the length of the forecast prognosis. The optimization problem is subject to minimizing the costs and CO_2 emissions of the schedule:

$$\min O_{cost}(S) = \sum_{i=1}^{n} cost_{gas,i} + cost_{el,i} + cost_{dh,i} - income_{el,i} \quad (3)$$

$$\min O_{CO2}(S) = \sum_{i=1}^{n} CO2_{gas,i} + CO2_{el,i} + CO2_{dh,i} \quad (4)$$

where $cost_{gas,i}$ is the price of gas consumed in the instant, $cost_{el,i}$ is the price of electricity consumed in the instant, and $cost_{dh,i}$ is the price of district heating consumed in the instant. The CHP produces electricity which needs to be deducted from $income_{el,i}$ in the instant. $CO2_{gas,i}$ is the emissions of gas consumed by the CHP and GB. $CO2_{el,i}$ is the emissions of electricity consumed by the HP. $CO2_{dh,i}$ *is* the emissions caused by district heating.

The optimization is subject to the following objective space constraints:

$$C_{electricity}(S) = \forall i \in S : P_{gh,i} + P_{e,hp,i} \leq P_{grid\_capacity} \quad (5)$$

$$C_{TES}(S) = \forall i \in S : T_{TES,min} \leq T_{TES,i} \leq T_{TES,max} \quad (6)$$

$C_{electricity}$(S) states that the greenhouse power demand and the HP power consumption must not exceed the grid capacity in any instant. C_{TES}(S) states that the temperature of the TES must be within the minimum and maximum values in any instant.

The optimization is subject to the following decision space constraints to reduce the size of the decision space.

$$C_{CHP}(S) = \forall C_i \in S : (0 \leq C_i \leq 1) \wedge (C_i \bmod C_r = 0) \quad (7)$$

$$C_{GB}(S) = \forall G_i \in S : (0 \leq G_i \leq 1) \wedge (G_i \bmod G_r = 0) \quad (8)$$

$$C_{DH}(S) = \forall D_i \in S : (0 \leq D_i \leq D_{max}) \wedge (D_i \bmod D_r = 0) \quad (9)$$

$$C_{HP}(S) = \forall H_i \in S : H_i = 0 \vee H_i = 1 \quad (10)$$

C_{CHP}(S) states that the load factor C_i must be between 0 and 1 while being divisible by the resolution C_r. C_{GB}(S) and C_{DH}(S) have similar properties but with separate resolutions G_r and D_r. C_{HP} state that the load factor H_i must be exactly 0 (turned off) or 1 (turned on).

The TES was configured with an initial temperature T_{init} to enable immediate flexible use. However, it must not be economically feasible to drain the TES completely in the last instants of the schedule as the optimization would regard the initial TES contents as costless energy. Therefore, an additional constraint was created to avoid this case:

$$C_{TES\_end\_temperature}(S) = \forall i \in S : T_{init} \leq T_{TES,\mathrm{n}} \quad (11)$$

where T_{init} is the initial temperature of the TES and $T_{TES,n}$ is the temperature in the last instant.

3.4 Scenario

The scenario is based on forecasts represented as several time series with hourly intervals. All input data were retrieved for the two-day period April 18th to April 19th, 2023.

The time series in the top of Fig. 5 shows the greenhouse's electricity and heat demands. The energy demands are based on optimal climate and historical energy consumption records (Qu, 2023).

The time series in the middle of Fig. 5 shows the energy prices for electricity, natural gas, and district heating. The electricity wholesale price signal was retrieved from Nord Pool in the Danish area DK1 and tariffs were retrieved from Energinet. The gas price signal was retrieved from the next day's index of the Exchange Transfer Facility (ETF) at the European Energy Exchange (EEX).

The time series in the bottom of Fig. 5 shows the CO2 emissions for electricity, natural gas, and district heating. The CO_2 emissions for electricity are available in a 5-minute resolution at Energinet. The 5-minute resolution was aligned with the other forecasts by aggregating the signal into an hourly resolution. CO_2 emissions for district heating are declared as a yearly average and were retrieved from the Danish provider Fjernvarme Fyn. The CO_2 emissions for gas are estimated to be 204 kg CO_2 / MWh by the Danish Energy Agency.

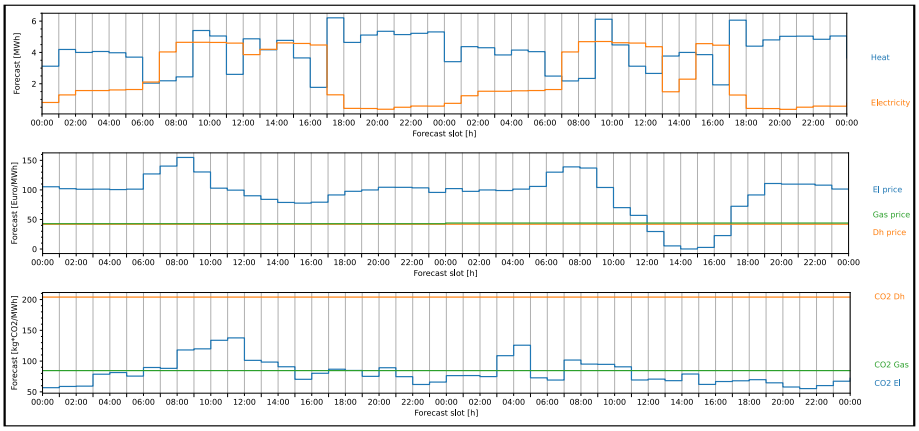

Fig. 5. The two-day forecasts used for the scenario.

4 Design and Implementation

4.1 Co-simulation implementation with mosaik

The closed-loop co-simulation was implemented in the mosaik co-simulation framework (Steinbrink et al., 2019). Mosaik provides the fundamental API and components for constructing an executable co-simulation of the case study including:

The Scenario script. The scenario script is a Python file responsible for instantiating simulators, configuring and connecting them through their exposed input/output attributes. Internally, mosaik constructs a data flow graph – a directed acyclic graph of the interdependent simulators including predecessors and successors.

The Component API. Mosaik assumes that a simulator is an independent piece of executable software that implements a simulation model. The component API is a simulator interface definition that mosaik uses to communicate with each simulator and coupling them through functions like `create`, `step`, and `getData`. Each simulator provides an implementation of the interface, and it may be exposed directly through Python (within the same process), or through JSON over TCP/IP (external simulation process). It is necessary to provide implementations of this interface if they do not already exist within the mosaik codebase. Mosaik provides implementations for e.g. FMI/FMU integration and exporting data in CSV format.

The Simulation manager. The simulation manager is responsible for establishing and maintaining the connections between mosaik and each simulator.

The Scheduler. The scheduler utilizes a master algorithm to coordinate the execution and data exchange of all simulators within the scenario. This master algorithm employs centralized coordination and synchronization mechanisms. Specifically, mosaik manages the global simulation time and instructs each simulator to advance accordingly,

subsequently retrieving the resulting data. Consequently, all data exchanges are mediated through mosaik.

The realized co-simulation is illustrated in a component diagram in Fig. 6. Each component, except the scheduler, provides an implementation of the mosaik Component API. From the diagram the following components are shown.

Scheduler. This component functions as described in the mosaik framework.

MOGA. This is the Java-based MOGA controller. The actual controller was adapted to be interoperable with mosaik by extending the abstract Java class `de.offis.mosaik.api.Simulator` and overriding its abstract methods `init(…)`, `create(…)`, `step(…)`, and `getData(…)`.

CSVWriter. This is a default component within the mosaik code base. It functions as an observer to export all relevant simulation into a CSV-file for later analysis.

Forecasts. This component contains a Python forecast simulation model that provides data in variable-sized sliding window as previously described in Section 3.1.

Local energy system. This package resembles the local energy system and contains FMI/FMU compatible simulation models of the CHP, GB, HP, DH, and TES. Mosaik provides an adapter to facilitate FMI/FMU compatibility.

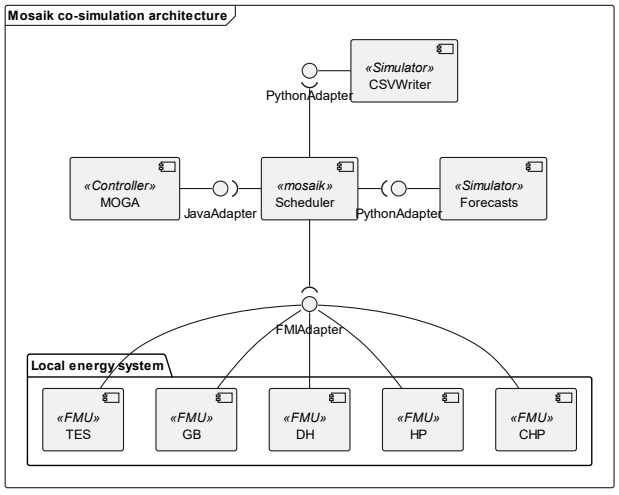

Fig. 6. The co-simulation composition in mosaik.

The simulators are synchronized on different timescales. The co-simulation's step size follows wall-clock time because the execution of the MOGA controller's termination criterion of 30 minutes. The forecasts and FMI/FMU simulators execute faster than their step sizes, so their outputs remain valid until the next simulation step. The FMI/FMU simulators have a 1-minute step size because their output values are collected at a 1-minute sample rate by the event-based CSVWriter component.

The first few synchronization points of the running example are shown in the timing diagram of Fig. 7.

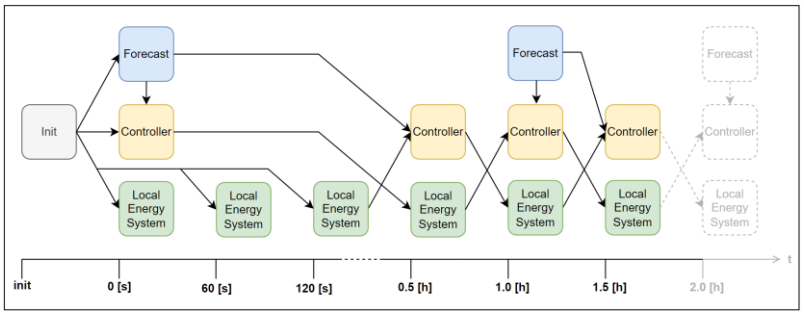

Fig. 7. A timing diagram showing the synchronization of each simulator in mosaik.

For simplicity, CSWWriter component is not shown, and all FMI/FMUs are aggregated inside the local energy system. Each simulator is configured as time-shifted to enable concurrent execution of the processes as would be the case in the real world. Because each simulator is time-shifted they must be initialized with initial input values. All simulators begin their execution path at time t = 0. The local energy system subsequently advances in 1-minute time steps, and the output values are collected by the CSVWriter. The closed-loop between the local energy system and the controller is established because the outputs of each simulator are used by the other in its next time step. This is indicated by the crossing zigzag data flow between the controller and the local energy system. The forecasts are updated hourly, and therefore new forecast data is fed to the controller at an hourly rate shown at t = 0 [s], t = 1 [h], t = 2 [h] in the diagram.

4.2 Practical design and implementation

A key aspect of the practical design and implementation is that it can serve as a skeletal system. NATS.io serves as the integration fabric which enables loosely coupled components to communicate in real-time through messaging. Components can be easily added to the infrastructure to extend capabilities. For instance, when the local energy system periodically publishes its state, it is straightforward to implement a logger component that stores the state for monitoring and analysis. Additionally, the local energy system exposes its state via request-response semantics when the controller requires immediate access. Fig. 8 shows the composition of the practical implementation where each component is connected to and communicates via NATS.io.

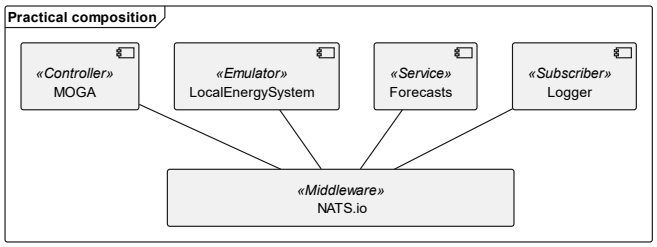

Fig. 8. The practical composition of the closed-loop control system.

The component boundaries are similar to those in the co-simulation. For instance, the controller remains an isolated component that uses models of the local energy system, its actual state, and forecast data to compute optimal setpoints. Similarly, the forecast data is treated as an isolated component that can be queried on demand. The local energy system is emulated as a single entity instead of separate entities (i.e., CHP, GB, HP, DH, and TES). The emulator exposes an endpoint that can receive setpoints for all these entities. This emulator can later be replaced with an adaptor that interacts with the real system via Programmable Logic Controllers (PLCs). Internally, the emulator simulates the behavior of the local energy system and periodically emits its state. All components are connected over a TCP/IP network and have their clock synchronized with the Network Time Protocol (NTP). The components reside on the same local area network (LAN), which reduces latency between them. The forecast component aggregates all relevant forecasts and exposes them uniformly over NATS.io. Messages are JSON formatted for readability but can use any other string or binary-compatible format e.g. Protocol Buffers, MessagePack, etc. Fig. 9 shows a combined sequence diagram of the interaction between the local energy system emulator and the controller.

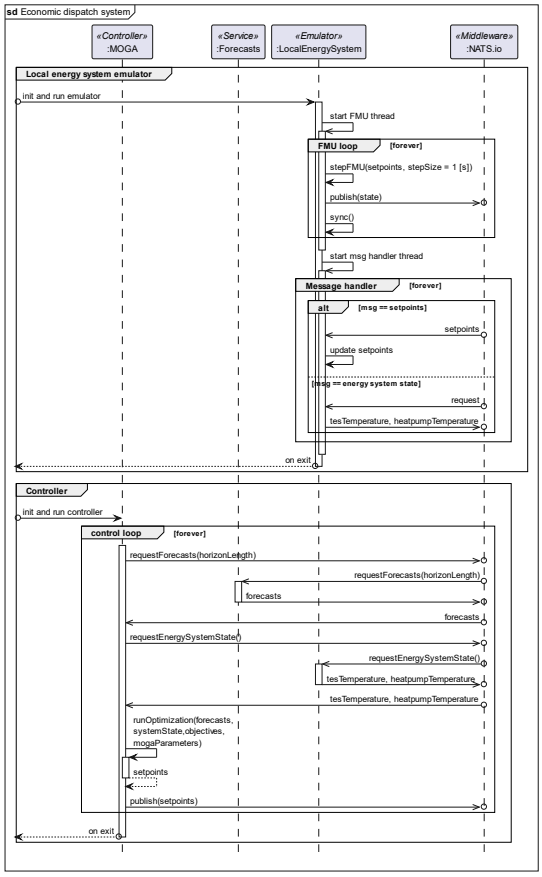

Fig. 9. Sequence diagram of the economic emission dispatch system.

The emulator is initialized with two concurrent threads. The first thread is responsible for continuously simulating the FMU and publishing the resulting state to NATS.io. The FMU is advanced in 1 second intervals, and the execution time of the FMU must be lower than 1 second to avoid drifting from wall-clock time. The `sync()` method simply blocks until 1 whole second has passed. The second thread listens for incoming messages and dispatches them accordingly. If the message contains a state request, the current state of the local energy system is returned. If the message contains setpoints, the internal FMU inputs are updated accordingly.

The controller executes within the first thread and contains an endless control loop. The control loop fetches the forecasts by sending a request through NATS.io that is forwarded to, and returned by, the forecasts service. The length of the forecasts varies accordingly to the variable-size sliding window as previously described. Then the current state of the local energy system is similarly requested through NATS.io. The MOGA optimization is then started given the parameters which are read from a configuration file. The `runOptimization(…)` method is a blocking call that returns when the termination criterion of 30 minutes has passed. The optimization returns the setpoints that were chosen based on the decision-making criteria which is the compromise between total energy costs and CO2 emissions in the Pareto-front. Finally, these setpoints are published to NATS.io which are then received by the local energy system emulator.

5 Results

This section presents the comparison of the two case study implementations. The comparison is based on an analysis of the functional behavior of the greenhouse energy system in terms of the energy output and TES temperature response. Both experiments were executed on a workstation with a 3.7 GHz (4.8 GHz Turbo) AMD Ryzen 9 5900X 12-core (24 threads) processor, 64 GB DDR4-3200 RAM, NVIDIA GeForce RTX 3060, and a Samsung 970 EVO NVMe drive. The experiments were configured with identical initial conditions in the local energy system, forecasts, and optimization parameters, including a fixed seed. Due to the real-time constraint of the MOGA and SIL, both experiments were executed in a duration of 24 wall-clock hours. The collected data output consists of two parts: (i) The output [MW] of each energy production unit for every 30-minute control interval, and (ii) the temperature over time within the TES. In the practical implementation, the network latency causes the controller to get slightly out of sync with the 30-minute interval with about one minute over 24 hours. This is explained by the MOGA termination criterion being set to 30 minutes which does not account for the added network latency to communicate with the other components. Therefore, the time dimension of the practical experiment output was aligned with that of the co-simulation by adding a column that assumes perfect synchronization between records.

Fig. 10 and Fig. 11 each show a stacked bar chart with the output [MW] per energy production unit at a 30-minute sample rate over 24 hours starting midnight April 18th and ending midnight April 19th. Accumulating the sampled MW output yields $s_{co\_sim} = 192.8$ and $s_{operational} = 192.3$, which means that both experiments produce a near equal

amount of energy within the period. Zooming in on the first hour (00:00 to 01:00), both experiments agree on the allocation. In the remaining hours the allocation is different in each interval, even when both experiments were configured with an identical seed. This behavior is revisited and explained in the end of this section. Most energy is allocated to district heating, which could indicate that this energy source is a good compromise between price and CO2 emissions. The heat pump is turned on for the entire duration which can be explained by its low CO2 emissions compared to the other energy sources. The CHP is mostly used in the hours 8:00 to 10:30 where the electricity prices and electricity demands are high. This indicate that it is economically feasible to use the CHP to satisfy the electricity demands and lowering the costs by selling the surplus electricity. The GB is the least used energy source in both experiments. However, the GB is allocated around 05:00, 14:00, and 17:00 to 19:30. At these times the gas and electricity CO2 emissions are almost equal, but the cost of gas is significantly lower.

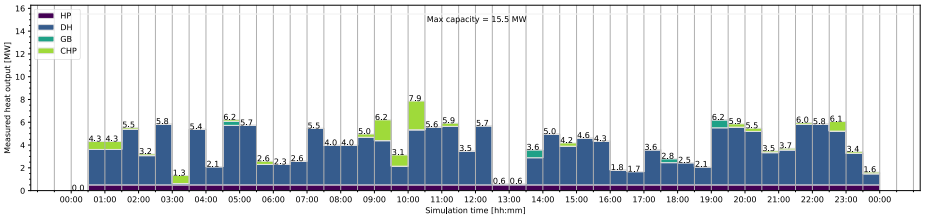

Fig. 10. April 18th production unit output [MW] of the co-simulation.

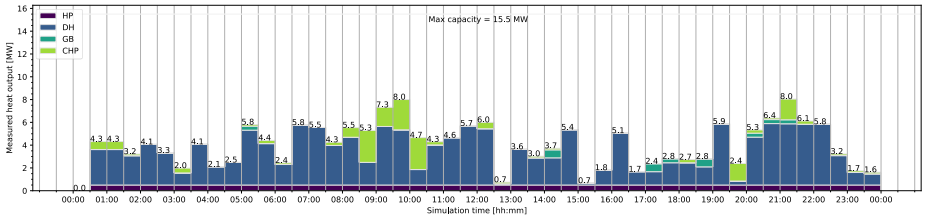

Fig. 11. April 18th production unit output [MW] of the practical implementation.

Fig. 12 shows the comparison of the TES temperature [°C] over the 24-hour period. Both experiments were configured with an initial TES temperature and end TES temperature of 50 °C. The end temperature (50 °C) in both experiments were satisfied. However, the TES utilization deviates over the 24-hour period. For example, the co-simulation experiment is reluctant to drain the TES from 00:00 to 08:30, where the operational experiment utilize energy from the TES. In both experiments, the temperature increases to about 57 °C at 12:30 and decreases towards 50 °C at the end of the experiment.

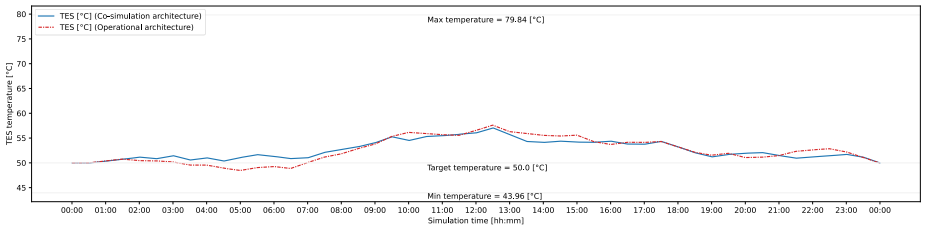

Fig. 12. A comparison of the temperature within the TES over time.

The behavior discrepancy of both energy output [MW] and temperature [°C] is caused by the deviating number of iterations (generations) performed by the MOGA within the 30-minute deadline. Even a slight variation can cause the two experiments to drift apart. This explanation was verified by modifying the termination criterion of the MOGA from a fixed time to a fixed number of iterations, which then yielded consistent results. Configuring a fixed number of iterations, however, cannot guarantee a 30-minute deadline. Such configuration can cause the controller to finish too early or too late, thus yielding unsatisfactory performance.

6 Discussion

This section discusses the challenges of transitioning the co-simulation implementation in mosaik to a practical implementation context.

Challenge 1: Transferring the scheduler. Mosaik's scheduler was not designed for operational environments and cannot be transferred directly. Consequently, all coordination including communication and synchronization between components must be achieved with other approaches or technologies. Practical communication approaches are often limited in expressing temporal semantics and therefore time becomes an emergent property of the implementation (Lee, 2018). As an example, the local energy system in the co-simulation advances its state as directed by the scheduler. In contrast, the practical implementation treats the local energy system as a representation of the physical process that advance time concurrently and thus require periodic sampling through a request/polling mechanism. Practitioners must carefully analyze the coordination of the co-simulation and choose implementation techniques that can satisfy these requirements without relying on the co-simulation scheduler.

Challenge 2: Transferring the co-simulation scenario script. The scenario script in mosaik provides a centralized and convenient way of initializing the co-simulation simulators. This includes configuration of parameters, initial conditions, connections, and exchanged input/output attributes. Transferring the scenario script into a practical context, requires careful analysis of both the scenario script and the involved simulators. Each simulator's adapter code must be analyzed because it is responsible for the synchronization behavior (time-triggered, event-based or hybrid). Depending on the size and complexity of the scenario, this task can be difficult and error prone. In this study, the scenario script was split into separate configurations deployed with each subsystem. For example, the controller contained MOGA-specific configurations, and the local

energy system contained the initial energy system conditions. Other techniques can be used, for example, centralized configuration management residing on a remote server that each subsystem query initially.

Challenge 3: Shifting away from the co-simulation protocol. In mosaik co-simulation, all simulators must adapt the underlying co-simulation protocol. The protocol consists of a basic set of operations e.g. `init()`, `create()`, `step()`, and `getData()`. The protocol uses the dependency inversion principle where each simulator awaits the step signal initiated by the co-simulator. A practical implementation will likely involve protocols that do not align with the semantics of these operations due to varying communication needs. This was not a huge issue in the case study; however, many smart grid co-simulations today involve complex protocols like the DNP3, IEC 61850, DLMS/ COSEM, and Open Charge Point Protocol (OCPP). For mosaik currently no adapters for those protocols are provided, but an adapter for IEC 60870 is available, which shows the option to extend the mosaik ecosystem in this direction in the future to facilitate the transition from co-simulation to real-world implementation. This approach requires implementing adapters for every protocol required in a scenario which impose more work.

Importance of this study: This study has identified and discussed some of the challenges in the transition process from co-simulation to practical implementation of economic emission dispatch in a greenhouse compartment. These challenges are important for practitioners to uncover to inform them in the realization of co-simulated energy systems. It was shown that the co-simulation could serve as a functional baseline for comparing the practical implementation. A co-simulation baseline is valuable because deviant behavior with the practical implementation can be detected early and guide the transitioning process. The literature on bridging the gap between co-simulation and real-world applications is sparse, and this paper contributes towards decreasing this gap and can hopefully spark interest for further research in this area.

Limitations and bias: The results and discussed challenges were deducted from transitioning a mosaik co-simulation to an architecture based around NATS.io. Consequently, technological and tool dependency bias is a risk for validity. Further studies are required to generalize the challenges to other domains and co-simulation technology. It is expected that the challenges should hold for cases where (i) the co-simulation technology employ a centralized scheduler (master algorithm), and (ii) there are dissimilarities between the co-simulation architecture and practical architecture that disallows a gradual transition towards field deployment.

7 Conclusion and Future Work

Co-simulation is often focusing on testing in an early stage of the development of new technologies, thus the literature on transitioning co-simulations into practical applications is limited. Challenges faced by practitioners in this process, must be identified, and solved before co-simulation can play a central role in whole development chain of new technologies from the prototyping to the real-world implementation. This paper contributes to this knowledge by transitioning a co-simulation of economic emission

dispatch into a practical implementation. This paper applied the principles from software-in-the-loop where the actual control software is used in both the co-simulation and practical implementation. This made the transitioning easier because only the interface between the control software and the local energy system needed to be rewritten. The internal economic emission dispatch algorithm remained unchanged. A few challenges were uncovered in this process, namely, (i) transferring the co-simulation scheduler and (ii) the co-simulation scenario script, and (iii) dealing with the shift away from the co-simulation protocol by using NATS.io to implement multimodal communication patterns. Despite of these challenges, co-simulation is still valuable because it can serve as a model and baseline that reflects correct behavior in an undisturbed environment. Practitioners can achieve confidence in their practical implementation by comparing it against the co-simulation baseline. Researchers and practitioners can use the work in this paper to inspire and inform the transitioning process. Many aspects still need to be researched. For example, (i) how to achieve faster-than wall-clock execution of the control software, (ii) assess if the co-simulation can be used as a baseline for evaluating non-functional requirements in the practical implementation, and (iii) developing a method that enables a more seamless transition process from early co-simulations into practical usable applications by also accounting for the operating conditions of the target system architecture.

Acknowledgements

This research is part of the Digital Energy Hub funded by the Danish Industry Foundation, the Project “IEA IETS Annex Task XVIII - Digitization, artificial intelligence and related technologies for energy efficiency and reduction of greenhouse gas emissions in industry” funded by EUDP (Case no.134-21010), and the Project "Danish Participation in IEA IETS Task XXI - Decarbonizing industrial systems in a circular economy framework", funded by EUDP (project number: 134233-511205).

Author contributions

Based on the CRediT (Contributor Roles Taxonomy). **CSBC**: Conceptualization, metodology, software, formal analysis, investigation, writing - original draft, Writing – review & editing, visualization. **SL**: Resources, Supervision. **JSS**: Resources, Supervision. **BNJ**: Supervision, writing- review and editing, project management. **ZGM**: Supervision, research management, funding acquisition.

References

Alleyne, A., Allgöwer, F., Ames, A., Amin, S., Anderson, J., Annaswamy, A., Antsaklis, P., Bagheri, N., Balakrishnan, H., & Bamieh, B. (2023). Control for Societal-scale Challenges: Road Map 2030. *IEEE Control Systems Society Publicat*.

Blochwitz, T., Otter, M., Åkesson, J., Arnold, M., Clauss, C., Elmqvist, H., Friedrich, M., Junghanns, A., Mauss, J., Neumerkel, D., Olsson, H., & Viel, A. (2012). *Functional Mockup*

Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. https://doi.org/10.3384/ecp12076173

Chowdhury, B. H., & Rahman, S. (1990). A review of recent advances in economic dispatch. *IEEE Transactions on Power Systems*, *5*(4), 1248-1259. https://doi.org/10.1109/59.99376

Clausen, C. S. B., Jørgensen, B. N., & Ma, Z. (2024a). A Modifiable Architectural Design for Commercial Greenhouses Energy Economic Dispatch Testbed. In B. N. Jørgensen, L. C. P. da Silva, & Z. Ma, *Energy Informatics* Cham.

Clausen, C. S. B., Jørgensen, B. N., & Ma, Z. G. (2024b). A scoping review of In-the-loop paradigms in the energy sector focusing on software-in-the-loop. *Energy Informatics*, *7*(1), 12. https://doi.org/10.1186/s42162-024-00312-8

Clausen, C. S. B., & Sørensen, J. V. (2021). Architectural Refinement of a Multi-Objective Multi-Issue Optimization Framework. In.

Coello, C. A. C., Lamont, G. B., & Veldhuizen, D. A. V. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. https://doi.org/10.1007/978-0-387-36797-2

Energinet. *Energi Data Service*. Retrieved June 25th from https://www.energidataservice.dk/

Gomes, C., Thule, C., Broman, D., Larsen, P. G., & Vangheluwe, H. (2018). Co-Simulation: A Survey. *ACM Comput. Surv.*, *51*(3), Article 49. https://doi.org/10.1145/3179993

Hardy, T. D., Palmintier, B., Top, P. L., Krishnamurthy, D., & Fuller, J. C. (2024). HELICS: A Co-Simulation Framework for Scalable Multi-Domain Modeling and Analysis. *IEEE Access*, *12*, 24325-24347. https://doi.org/10.1109/ACCESS.2024.3363615

Hassan, M. H., Yousri, D., Kamel, S., & Rahmann, C. (2022). A modified Marine predators algorithm for solving single- and multi-objective combined economic emission dispatch problems. *Computers & Industrial Engineering*, *164*, 107906. https://doi.org/https://doi.org/10.1016/j.cie.2021.107906

IEA. (2023). *Unlocking Smart Grid Opportunities in Emerging Markets and Developing Economies*. https://www.iea.org/reports/unlocking-smart-grid-opportunities-in-emerging-markets-and-developing-economies

Lee, E. A. (2018). What Is Real Time Computing? A Personal View. *IEEE Design & Test*, *35*(2), 64-72. https://doi.org/10.1109/MDAT.2017.2766560

Mahdi, F. P., Vasant, P., Kallimani, V., Watada, J., Fai, P. Y. S., & Abdullah-Al-Wadud, M. (2018). A holistic review on optimization strategies for combined economic emission dispatch problem. *Renewable and Sustainable Energy Reviews*, *81*, 3006-3020. https://doi.org/https://doi.org/10.1016/j.rser.2017.06.111

Maniatopoulos, M., Lagos, D., Kotsampopoulos, P., & Hatziargyriou, N. (2017). Combined control and power hardware in-the-loop simulation for testing smart grid control algorithms. *IET Generation, Transmission & Distribution*, *11*(12), 3009-3018. https://doi.org/https://doi.org/10.1049/iet-gtd.2016.1341

Monostori, L. (2014). Cyber-physical Production Systems: Roots, Expectations and R&D Challenges. *Procedia CIRP*, *17*, 9-13. https://doi.org/https://doi.org/10.1016/j.procir.2014.03.115

NordPoolGroup. *Nord Pool | Day-ahead prices*. Retrieved June 25th from https://data.nordpoolgroup.com/auction/day-ahead/prices

Ofenloch, A., Schwarz, J. S., Tolk, D., Brandt, T., Eilers, R., Ramirez, R., Raub, T., & Lehnhoff, S. (2022, 4-5 April 2022). MOSAIK 3.0: Combining Time-Stepped and Discrete Event Simulation. 2022 Open Source Modelling and Simulation of Energy Systems (OSMSES),

Qu, Y. (2023). *A Digital Twin Framework for Commercial Greenhouse Climate Control System* University of Southern Denmark (The Maersk Mc Kinney Moller Institute)].

Rizk-Allah, R. M., Hagag, E. A., & El-Fergany, A. A. (2023). Chaos-enhanced multi-objective tunicate swarm algorithm for economic-emission load dispatch problem. *Soft Computing*, *27*(9), 5721-5739. https://doi.org/10.1007/s00500-022-07794-2

Samad, T., Bauer, M., Bortoff, S., Di Cairano, S., Fagiano, L., Odgaard, P. F., Rhinehart, R. R., Sánchez-Peña, R., Serbezov, A., Ankersen, F., Goupil, P., Grosman, B., Heertjes, M., Mareels, I., & Sosseh, R. (2020). Industry engagement with control research: Perspective and messages. *Annual Reviews in Control*, *49*, 1-14. https://doi.org/10.1016/j.arcontrol.2020.03.002

Sharvari, T., & Sowmya, N. K. (2019). A study on Modern Messaging Systems- Kafka, RabbitMQ and NATS Streaming. In. Ithaca.

Steinbrink, C., Blank-Babazadeh, M., El-Ama, A., Holly, S., Lüers, B., Nebel-Wenner, M., Ramírez Acosta, R. P., Raub, T., Schwarz, J. S., Stark, S., Nieße, A., & Lehnhoff, S. (2019). CPES Testing with mosaik: Co-Simulation Planning, Execution and Analysis. *Applied Sciences*, *9*(5). https://doi.org/10.3390/app9050923

Van Der Meer, A. A., Palensky, P., Heussen, K., Bondy, D. E. M., Gehrke, O., Steinbrinki, C., Blanki, M., Lehnhoff, S., Widl, E., Moyo, C., Strasser, T. I., Nguyen, V. H., Akroud, N., Syed, M. H., Emhemed, A., Rohjans, S., Brandl, R., & Khavari, A. M. (2017). Cyber-physical energy systems modeling, test specification, and co-simulation based testing. *2017 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems, MSCPES 2017 - Held as part of CPS Week, Proceedings*. https://doi.org/10.1109/MSCPES.2017.8064528

Vogt, M., Marten, F., & Braun, M. (2018). A survey and statistical analysis of smart grid co-simulations. *Applied Energy*, *222*, 67-78. https://doi.org/https://doi.org/10.1016/j.apenergy.2018.03.123

Yang, T. (2022). Analysis and Application of Model Predictive Control in Energy Systems. In: Syddansk Universitet. Det Tekniske Fakultet.