3rd NFDI4Energy Conference: Defining Energy Research Data Management Together
Proceedings to the 3rd NFDI4Energy Conference - Full Papers
https://doi.org/10.52825/ocp.v9i.3271

Published: 23 Mar. 2026

Reusable Agent-Based Simulations of Cyber-Physical Energy Systems: Environments as First-Class Entities

Rico Schrage[1,2,*] and Astrid Nießе[1,2]

[1]Carl von Ossietzky Universität Oldenburg, Oldenburg, Germany
[2]OFFIS - Institute for Information Technology, Oldenburg, Germany
*Correspondence: Rico Schrage, rico.schrage@uni-oldenburg.de

Abstract. Agent-based simulation in cyber-physical energy systems is a challenging task, often involving multiple dimensions and layers, heterogeneous actors, and large-scale systems, to serve as an effective analysis tool. It is also hard to compare multi-agent systems because most simulations use complex, non-reusable agent environments. To address these problems, a simulation framework that supports not only complex simulation but also reusable environments for agent systems is needed. This paper describes a Julia-based simulation framework for agent-based simulations of cyber-physical (energy) systems, featuring an environment API that enables the creation of reusable environments. For this purpose, an event-based architecture for agent-environment interaction is proposed, implemented in a framework, and demonstrated as a showcase.

Keywords: FAIR4RS, Agent-Based Simulation, Cyber-Physical Energy Systems, Reusable Simulations, Agent Framework, Energy System Optimization

1. Introduction

Agent-based modeling (ABM) provides a powerful framework for representing the decentralized behavior of actors in energy systems. These models are naturally complex, as multiple actors operate on different parts of the system, often in spatially separated areas, and represent a variety of stakeholders (e.g., system operator, energy market operator, private or industrial customers). Popular applications include the modeling of electricity markets, energy trading, and distributed grid control [1], [2].

To cope with the variety of applications, decentralized and asynchronous agent behavior, and to provide a well-engineered technical foundation for execution, agent-based simulation (ABS) frameworks have emerged. We refer to agent-based simulation as the computational implementation of an agent-based model, including explicit mechanisms for time management, agent scheduling, and interaction handling. We define an agent as a software unit that observes an environment through sensors and acts upon it using actuators [3].

In an agent-based simulation framework for cyber-physical energy systems, it is important to be able to describe agent behavior and the interactions between agents via

communication in a simple way. Another, often-overlooked aspect, is the description of the environment in which the agents live. While introducing general environments as a separate unit type poses some challenges, it also creates the opportunity to design environments that enable actual reusability and comparability. Emphasizing this also aligns with the FAIR4RS principles [4], [5], particularly interoperability and reusability, and has been advocated by Ferenz et al. [6] in the context of standardizing energy research software.

The landscape of existing agent-based simulation frameworks is dense but still heterogeneous, with many engineered for different methodological purposes (e.g., reinforcement learning agents, language model agents, ...). Our framework combines the idea of reusable environments (already available for specific methods like Reinforcement Learning (RL)) with the capabilities of a general-purpose, communication-based agent-based simulation framework. Therefore, it provides a solution to the increasing complexity of simulating energy systems by enabling the reuse of system simulations. This simplifies the comparison of agent-based (or generally distributed) approaches in energy research.

This paper is structured as follows. First, we further motivate the research gap in the related work section. After that, the requirements for the frameworks are formulated. The architecture is then described and explained. Next, the simulation methodology and the implementation are briefly introduced. At last, a short showcase is presented, and a conclusion is drawn.

2. Related Work

Agent-based modeling and simulation is a well-established methodology in energy system research and is applied across a wide range of domains, including electricity markets, demand response, intelligent grid control, and multi-energy systems [1]. The growing complexity of cyber-physical energy systems, however, has exposed limitations in existing simulation frameworks, particularly in reusability, comparability, and communication integration. This section reviews related work with a focus on agent-based simulation frameworks, energy-system co-simulation approaches, and reusable environment concepts.

Agent-based simulation frameworks

General-purpose agent-based modeling (ABM) frameworks such as NetLogo [7], Mesa [8], and Agents.jl [9] provide mature toolchains for defining agents, their interactions, and simulation scheduling. These frameworks typically offer abstractions for agents, global state, and spatial representations such as grids, networks, or continuous spaces.

In most ABM frameworks, the environment is implicitly represented in the shared simulation state or explicitly encoded in agent logic. While this approach offers great modeling flexibility, it tightly couples the environment to a specific simulation implementation. As a result, environments are rarely reusable across different agent behaviors or experimental configurations. The lack of a standardized agent-environment interaction interface makes it difficult to compare different distributed control strategies or agent implementations on the same system model. Further, in these frameworks, network communication is not supported.

However, the communication-aware framework mango [10], designed for cyber-physical energy systems, focuses on real-time communication and does not explicitly

address environment abstractions or comprehensive simulation-time support. There are also specific agent-based frameworks for energy systems, such as ASSUME [2], and SIMONA [11], which provide energy-specific tools, environments, and methods. These frameworks are implemented with a different abstraction level in mind and therefore fall into a different category.

Energy-system co-simulation and cyber-physical modeling

To cope with the complexity of modern energy systems, a variety of co-simulation frameworks have been developed. Tools such as mosaik [12] and HELICS [13] allow coupling heterogeneous simulators that represent different physical, control, and communication domains. The mosaik framework and HELICS focus on orchestrating multiple simulators with different time semantics and are widely used in smart-grid and cyber-physical energy-system studies.

While co-simulation frameworks enable detailed multi-domain modeling at scale, they do not expose a unified agent-centric environment abstraction. Agents interact with simulator-specific APIs, and the environment remains fragmented across coupled components. This limits reusability and complicates systematic comparison of different agent-based approaches.

Reusable environments and reinforcement learning

Reinforcement learning has introduced a standardized notion of reusable environments through libraries such as Gymnasium [14]. These environments define a clear agent–environment interaction interface, decoupling agent logic from system dynamics and enabling reproducible benchmarking. Energy-specific RL environments such as Grid-2-Op [15], CityLearn [16], and ofpgym [17] demonstrate the benefits of this approach by allowing different control strategies to be evaluated on identical system models. However, these environments are fundamentally shaped by reinforcement-learning assumptions, including synchronous step-based interaction and reward-centric evaluation.

Discussion and research gap

The reviewed literature highlights a gap between reusable environment abstractions and general-purpose agent-based simulation for cyber-physical energy systems. Existing ABM frameworks lack explicit, reusable environment concepts; energy-system co-simulation tools expose environments only implicitly; and RL environments impose learning-specific interaction models.

This gap motivates the need for a reusable agent-environment abstraction that is independent of specific agent methodologies and integrates communication into the environment's dynamics. The framework proposed in this work addresses this need by generalizing the concept of environments beyond reinforcement learning and embedding it in a communication-aware agent-based simulation architecture for cyber-physical energy systems.

3. Requirements

To clarify our development objective, we will outline the requirements for developing the framework. For agent-based simulations, several key points need to be considered. By definition, cyber-physical systems consist of a physical system and a cyber system

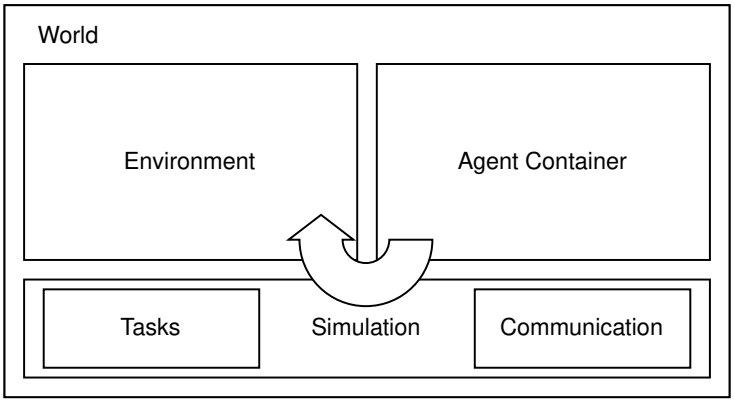


Figure 1. *Architecture of the agent-based simulation*

(the communication system to coordinate the system). Given the balancing objective of demand and supply, time is an important factor. Further, energy system simulations can be computationally intensive, as significant time and spatial spans may need to be incorporated. Further energy systems are part of the critical infrastructure; consequently, system resilience is one of the most important aspects, underscoring the need to include communication incidents and, therefore, communication simulation as an integral part of the simulation. Due to the need for reproducibility and reusability [18], [19], this framework has been developed with these properties as its primary focus. To incorporate these requirements, the following specific aspects are additionally considered.

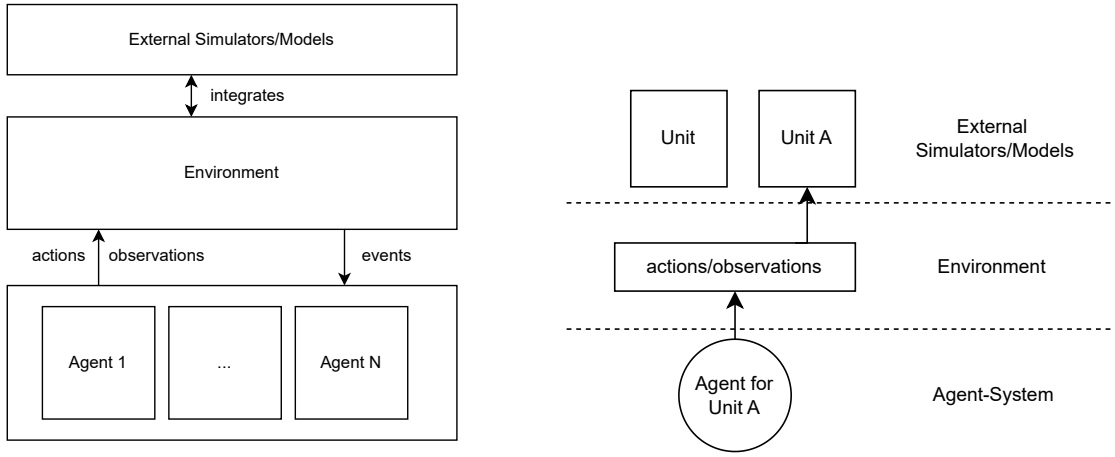
1. To consider realistic conditions for the agent-based actions, communication delays are included. For this reason alone, it is necessary to track a simulation time.
2. The framework must provide deterministic, well-defined task ordering per entity.
3. The framework must support stepping external simulators in sync with the discrete-event clock (and optionally at configured update points).
4. As research should be reproducible and comparable, we aim to separate the agent-system and the environment (physical simulation) from each other. This enables other researchers to reuse the environment to implement their own agent-based approach.
5. Problems in the field of energy research are heterogeneous; therefore, a general-purpose framework shall be developed to apply to various problems.

4. Architecture

To support reusable environments as first-class citizens and include communication simulation, the architecture needs to be designed accordingly. Most importantly, the interface between the agent system and the environment needs to be well-defined, support the implementation of arbitrary environments, and ensure that the interaction with the environment remains manageable and accessible to the agents.

In the following, we describe the architecture, differentiating between two main entities: the agent and the environment. However, the capsule of a set of agents, the agent container, is the environment's architectural counterpart, being on the same architectural level. The container and the environment are part of the so-called world. The world also includes task and communication simulators and schedulers that execute the environment and agents' tasks correctly (see Figure 1).

We first start with the agent, then proceed to the environment, and finally describe the simulation module.



(a) *Architecture of the interaction models between environment and agent system*

(b) *Role of the environment for retrieving actions and observations as an agent*

Figure 2. *Interaction of the agent systems and the environments with their encapsulated external simulators and models.*

4.1 Agent

An agent is part of an agent system in an agent container (see Figure 1). An agent can send messages, handle messages, schedule tasks (periodic, delayed, timestamp tasks, ...), observe the environment, and take actions on the environment (see Figure 2b).

Agents can be part of multiple topologies, which can be expressed as a connectivity graph $G = (V, E)$ with V as the set of agents $\{0, 1, \ldots, n\}$, and the $n \times n$ adjacency matrix A, with a_{ij} (with $i, j \in V$) is 1 if i and j are connected, and 0 if there is no connection. The framework provides methods for creating these topologies and distributing the neighborhoods.

4.2 Environment

The environment integrates all non-agent units that participate in the simulation. In energy systems, it will mainly contain physical components (e.g., distributed energy resources) or actors that do not require agent representation (e.g., markets).

The environment consists of a behavior and a space. The behavior determines the environment's stepping logic (e.g., updating models, moving actors, notifying agents, ...). The space represents the physical space in which the agents exist. Agents can retrieve and update their own positions, and other agents can also be discovered. The space is generally optional, as agents can be stationary.

It provides each agent with observations and actions. To achieve this, an agent can be *installed* in an environment using some identifier (arbitrary) which can identify the component/part of the physical system the agent should be assigned to (see Figure 2). To interact with agents, the environment can emit global and agent events (received in the same step) that indicate that something happened. Agent events are emitted to some identifier of an installation. All agents installed under that identifier are automatically subscribed to these events. Agent events represent the scenario of a local device sending a local signal to its representative. Further, agents can freely subscribe to

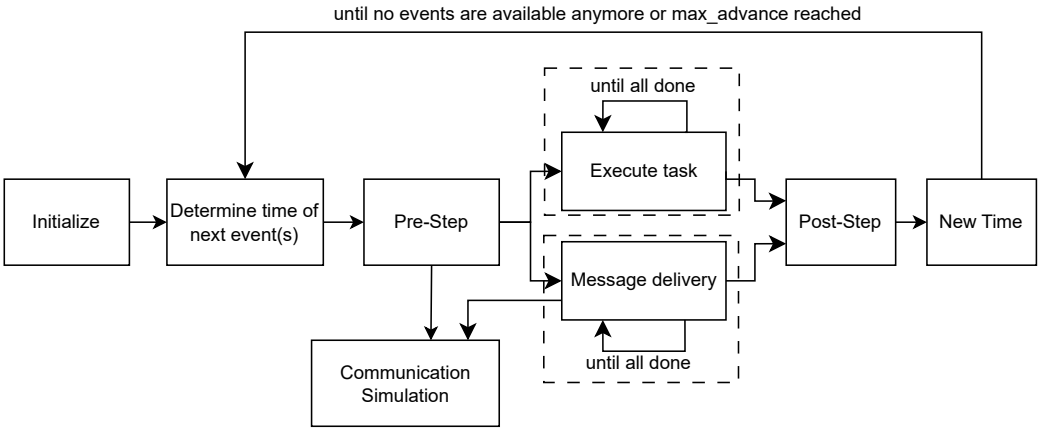


Figure 3. *Architecture of the agent-based simulation*

global events. These event types represent changes that can be globally observed (e.g., weather changes, system blackouts, ...).

4.3 Simulation Flow

The simulation integrates the agent system (as an agent container) and the environment within a single simulation loop. The world implements this simulation with two event-executing loops: the task and the communication loop. To queue tasks, agents and the environment can create tasks and add them to the event queue. The general execution flow of one step of the simulation can be expressed as follows.

1. Initialization of the environment.
2. Determine the next events $\mathcal{E}$ and calculate the step size.
3. Step all entities (environment and agents)
4. Execute all communication and task events $\in \mathcal{E}$ (parallelized).
5. Update the internal simulation time
6. Return the results of the step

The full flow is visualized in Figure 3. Note that there can be pre- and post-steps, e.g., for data recording during simulation. This simulation flow shows the implementation of the discrete-event simulation. However, the framework also supports continuous time simulation. In that case, the step size can be used as input to the simulation.

5. Implementation

The framework, environment extension, agent model, and simulation methodology are implemented at https://github.com/OFFIS-DAI/Mango.jl, which also supports real-time simulation using communication standards such as TCP and MQTT. The Julia language was chosen due to the large ecosystem [20], [21], [22], its focus on simplicity for scientific applications, and its easily applicable, mature threading API (enabling high core utilization).

6. Showcase

To demonstrate the frameworks' and the architectures' capabilities and usefulness, we present a specific example environment. We aim to evaluate two agent systems using

```
# For each agent to install it on the environment, enables agent events
...
install(world.env, agent, id=get_uuid(component), type=:component)
...

# Agent is notified when the local power demand changes
function on_agent_event(agent, clock, event)
    # Agent can access installed observations
    power = observation(agent, :max_active_power)
    # Notifies the demand aggregator
    send_message(agent, (power, clock.simulation_time), agent.target)
end
```

Figure 4. *Implementation of the interaction between agents and the environment (simplified)*

the exact same environment, solving the economic dispatch problem (minimizing the cost while meeting the demand target and staying within the power limits). The first agent system uses an averaging consensus algorithm (see [23]). The second monitors devices using agents and sends all necessary data to an aggregator agent, which solves the problem centrally using linear programming. Economic dispatch must be executed every 15 minutes due to changing demand. Additionally, we want to validate whether the approaches are robust to communication incidents (here, packet losses).

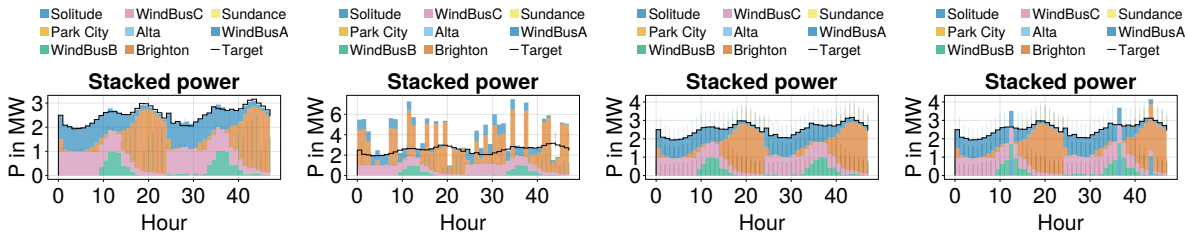
To achieve this, PowerSystems.jl [21] is integrated as the core driver of an environment for an agent system. PowerSystems.jl can provide datasets and components for scenarios. To integrate this external simulator, the first step is to identify which units may be controlled by agents. Second, we determine the observations and actions for each unit. In PowerSystems.jl, the primary entity is a *component*. Each component has an identifier; therefore, each agent can be assigned a component by assigning it the component's identifier. This identifier is later used to install the agent in the environment, enabling it to observe and act on the specific component after the mapping is done.

Because the agent simulation is time-dependent, the time-series data for each component (e.g., a solar power plant schedule) is applied at the correct simulation time. The environment's behavior handles this. The full environment and the agent system are published on GitHub (see Underlying and related material).

The exemplary results of this showcase are shown in Figure 5. The exemplary results show that the consensus implementation is more sensitive to packet losses than the central agent. The higher loss rate for the central agent system was used to make an effect visible. It also shows that the consensus performance is close to the central agent's performance in the lossless case.

7. Conclusion

This work introduced an agent-based simulation framework for cyber-physical energy systems that promotes environments to first-class, reusable entities. By formalizing an agent-environment interaction interface and integrating communication, the framework addresses key challenges in existing simulation tools. The proposed architecture decouples physical system models from agent logic, enabling systematic comparison of distributed control strategies and improving the reproducibility of energy-system studies by enabling the capability to develop FAIR-compliant simulations for agent-based approaches.



(a) *Consensus results without packet losses.* ***(b)*** *Consensus results with 0.005%.* ***(c)*** *Central agent results without packet losses.* ***(d)*** *Central agent results with 0.5% losses.*

Figure 5. *Stacked area graph for the power contribution in MW. The solid black line depicts the demand. The labels correspond to the scenario data from the PowerSystemCaseBuilder.jl [24] scenario used. Loss rates differ across approaches to visualize effects; lossless cases provide the direct baseline comparison.*

The Julia-based implementation demonstrates how simulators can be encapsulated within environments while providing agents with observations and actions. The showcase further highlights how the framework enables realistic, reusable simulations. This underlines the framework's suitability for evaluating distributed algorithms under realistic cyber-physical conditions.

Future work will further extend the library of reusable environments, and analyze and improve the actual simulation performance.

Data availability statement

The original data shown here is published on https://doi.org/10.5281/zenodo.18636100.

Underlying and related material

The underlying framework is available at https://github.com/OFFIS-DAI/Mango.jl. The environment and the showcase are available at https://github.com/Digitalized-Energy-Systems/MangoEnergyEnvironments.jl and, https://github.com/Digitalized-Energy-Systems/EnergySchedulingBenchmark.jl.

Author contributions

R. Schrage is responsible for the Conceptualization, the Methodology, the Software, and the Writing - original draft. A. Nießе is responsible for Supervision, and Writing - review & editing.

Competing interests

The authors declare that they have no competing interests.

Funding

This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 359941476.

References

[1] P. Ringler, D. Keles, and W. Fichtner, "Agent-based modelling and simulation of smart electricity grids and markets – a literature review," *Renewable and Sustainable Energy Reviews*, vol. 57, pp. 205–215, 2016, ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2015.12.169. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S136403211501552X.

[2] N. Harder et al., "Assume: An agent-based simulation framework for exploring electricity market dynamics with reinforcement learning," *SoftwareX*, vol. 30, p. 102 176, 2025, ISSN: 2352-7110. DOI: https://doi.org/10.1016/j.softx.2025.102176. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352711025001438.

[3] S. J. Russell and P. Norvig, *Artificial intelligence a modern approach*. London: Prentice Hall, 2010.

[4] N. P. Chue Hong et al., *Fair principles for research software (fair4rs principles)*, version 1.0, May 2022. DOI: 10.15497/RDA00068. [Online]. Available: https://doi.org/10.15497/RDA00068.

[5] M. Barker et al., "Introducing the fair principles for research software," *Scientific Data*, vol. 9, no. 1, p. 622, 2022. DOI: https://doi.org/10.1038/s41597-022-01710-x.

[6] S. Ferenz et al., "Ten recommendations for engineering research software in energy research," in *Proceedings of the 16th ACM International Conference on Future and Sustainable Energy Systems*, ser. E-Energy '25, Association for Computing Machinery, 2025, pp. 446–459, ISBN: 9798400711251. DOI: 10.1145/3679240.3734606. [Online]. Available: https://doi.org/10.1145/3679240.3734606.

[7] U. Wilensky, "NetLogo," *Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL*, 1999. [Online]. Available: https://ccl.northwestern.edu/netlogo/.

[8] E. ter Hoeven et al., "Mesa 3: Agent-based modeling with Python in 2025," *Journal of Open Source Software*, vol. 10, no. 107, p. 7668, Mar. 2025. DOI: 10.21105/joss.07668. [Online]. Available: https://joss.theoj.org/papers/10.21105/joss.07668.

[9] G. Datseris, A. R. Vahdati, and T. C. DuBois, "Agents.jl: A performant and feature-full agent-based modeling software of minimal code complexity," *SIMULATION*, vol. 0, no. 0, p. 003 754 972 110 688, Jan. 2022. DOI: 10.1177/00375497211068820. [Online]. Available: https://doi.org/10.1177/00375497211068820.

[10] R. Schrage, J. Sager, J. P. Hörding, and S. Holly, "Mango: A modular python-based agent simulation framework," *SoftwareX*, vol. 27, p. 101 791, 2024, ISSN: 2352-7110. DOI: https://doi.org/10.1016/j.softx.2024.101791. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352711024001626.

[11] ie³ - Institute of Energy Systems, Energy Efficiency and Energy Economics - TU Dortmund University et al., *SIMONA - A Discrete-Event Distribution Grid Simulation Environment*, version 4.0.0, May 2025. [Online]. Available: https://github.com/ie3-institute/simona.

[12] C. Steinbrink et al., "Cpes testing with mosaik: Co-simulation planning, execution and analysis," *Applied Sciences*, vol. 9, no. 5, 2019, ISSN: 2076-3417. DOI: 10.3390/app9050923. [Online]. Available: https://www.mdpi.com/2076-3417/9/5/923.

[13] T. D. Hardy, B. Palmintier, P. L. Top, D. Krishnamurthy, and J. C. Fuller, "Helics: A co-simulation framework for scalable multi-domain modeling and analysis," *IEEE Access*, vol. 12, pp. 24 325–24 347, 2024. DOI: 10.1109/ACCESS.2024.3363615.

[14] M. Towers et al., *Gymnasium: A Standard Interface for Reinforcement Learning Environments*. [Online]. Available: https://github.com/Farama-Foundation/Gymnasium.

[15] B. Donnot, *Grid2op- A testbed platform to model sequential decision making in power systems.* 2020. [Online]. Available: %5Curl%7Bhttps://GitHub.com/Grid2Op/grid2op%7D.

[16] K. Nweye et al., "Citylearn v2: Energy-flexible, resilient, occupant-centric, and carbon-aware management of grid-interactive communities," *Journal of Building Performance*

Simulation, vol. 0, no. 0, pp. 1–22, 2024. DOI: 10.1080/19401493.2024.2418813. [Online]. Available: https://doi.org/10.1080/19401493.2024.2418813.

[17] T. Wolgast and A. Nieße, "Learning the optimal power flow: Environment design matters," *Energy and AI*, vol. 17, p. 100 410, 2024, ISSN: 2666-5468. DOI: https://doi.org/10.1016/j.egyai.2024.100410. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666546824000764.

[18] R. Morrison, "Energy system modeling: Public transparency, scientific reproducibility, and open development," *Energy Strategy Reviews*, vol. 20, pp. 49–63, 2018, ISSN: 2211-467X. DOI: https://doi.org/10.1016/j.esr.2017.12.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2211467X17300949.

[19] G. M. Huebner, M. J. Fell, and N. E. Watson, "Improving energy research practices: Guidance for transparency, reproducibility and quality," *Buildings and Cities*, Jan. 2021. DOI: 10.5334/bc.67.

[20] A. Plietzsch et al., "Powerdynamics.jl–an experimentally validated open-source package for the dynamical analysis of power grids," *SoftwareX*, vol. 17, p. 100 861, 2022.

[21] J. D. Lara, C. Barrows, D. Thom, D. Krishnamurthy, and D. Callaway, "PowerSystems.jl — A power system data management package for large scale modeling," *SoftwareX*, vol. 15, Jul. 2021. DOI: https://doi.org/10.1016/j.softx.2021.100747. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352711021000765.

[22] C. Coffrin, R. Bent, K. Sundar, Y. Ng, and M. Lubin, "Powermodels.jl: An open-source framework for exploring power flow formulations," in *2018 Power Systems Computation Conference (PSCC)*, Jun. 2018, pp. 1–8. DOI: 10.23919/PSCC.2018.8442948.

[23] L. Jian, Z. Qian, Z. Liangang, and Y. Mengkai, "Distributed economic dispatch method for power system based on consensus," *IET Renewable Power Generation*, vol. 14, no. 9, pp. 1424–1432, 2020. DOI: https://doi.org/10.1049/iet-rpg.2019.1085. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-rpg.2019.1085. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-rpg.2019.1085.

[24] NREL-Sienna contributors, *PowerSystemCaseBuilder.jl, Package to build cases for power systems modeling*, version 2.1.0, GitHub repository, National Renewable Energy Laboratory (NREL), Nov. 30, 2025. Accessed: Dec. 15, 2025. [Online]. Available: https://github.com/NREL-Sienna/PowerSystemCaseBuilder.jl.