Simple and Flexible Power System Simulation Coupling using Remote Object Communication

1st Arjun Kumar Madhusoodhanan
Institute of Automation and Applied Informatics, Karlsruhe Institute of Technology,
Karlsruhe, Germany
arjun.madhusoodhanan@kit.edu

2nd Julian Hoffmann
Institute of Automation and Applied Informatics, Karlsruhe Institute of Technology,
Karlsruhe, Germany
julian.hoffmann@kit.edu

3rd Emmanuel Kalathingal Raj
Institute of Automation and Applied Informatics, Karlsruhe Institute of Technology,
Karlsruhe, Germany
upxcg@student.kit.edu

4th Steeve Brite Jose
Institute of Automation and Applied Informatics, Karlsruhe Institute of Technology,
Karlsruhe, Germany
ushnp@student.kit.edu

5th Uwe Kühnapfel
Institute of Automation and Applied Informatics, Karlsruhe Institute of Technology,
Karlsruhe, Germany
uwe.kuehnapfel@kit.edu

6th Veit Hagenmeyer
Institute of Automation and Applied Informatics, Karlsruhe Institute of Technology,
Karlsruhe, Germany
veit.hagenmeyer@kit.edu

2025 IEEE Kiel PowerTech | 979-8-3315-4397-6/25/$31.00 ©2025 IEEE | DOI: 10.1109/PowerTech59965.2025.11180629

***Abstract*—Simulation coupling in power systems is essential for addressing complex, multidisciplinary challenges by integrating specialized models from diverse domains. However, existing approaches face limitations such as data privacy concerns, security risks, hardware and licensing constraints, and the absence of standardized exchange formats and interfaces. This paper presents a simple, flexible, and secure method for simulation coupling using remote object communication. The implementation leverages Pyro (Python Remote Objects) to expose objects that facilitate simulation control and data transfer to create a distributed simulation that keeps the model confidential. Unlike resource-intensive co-simulation frameworks, this method provides lightweight integration with minimal programming and configuration overhead. We demonstrate the feasibility of this approach through a case study that couples DIgSILENT PowerFactory — a widely used power system simulation tool — via its API with Pyro for remote simulation execution and model parametrization. The results highlight the effectiveness of this approach in achieving seamless model integration, secure data exchange, and efficient distributed simulation for power system studies.**

***Keywords*—Power system modeling, Simulation coupling, Distributed simulation, Model Privacy, Secure data exchange, Pyro, Powerfactory.**

I. Introduction

Simulation models are valuable tools in the electrical energy system design, as they enable detailed analysis and optimization of increasingly complex processes. In particular, the ongoing energy transition —characterized by a growing share of decentral renewable energy sources, expanding energy storage capacities, and auxiliary energy infrastructure — relies heavily on robust simulation frameworks [1], [2]. As power systems become more and more interconnected with electricity markets, weather and climate data, communication networks, cyber systems, and gas grids for stable, reliable, efficient, and resilient operation, integrating models from various power system domains is essential to capture multi-domain interactions and achieve more accurate simulation results [3]. Simulation coupling is an effective approach for addressing multi-disciplinary problems by combining the strengths of different simulation tools and models, leading to a more comprehensive understanding of their interdependencies. In doing so, it facilitates collaboration among experts from different fields and can provide access to models without requiring the need for duplicated infrastructure, such as high-performance computing (HPC) and external databases or licenses required for certain tools, making it a cost-effective solution for solving complex problems [4]. With the growing demand for real-time decision-making in applications such as digital twins [5], hardware-in-the-loop testing, and cyber-physical systems [6], simulation coupling provides a scalable solution for integrating multi-domain models and accelerating the development of sustainable technologies for the energy system.

A. Motivation

While simulation coupling across different domains in energy systems offers significant potential for solving complex, multi-disciplinary problems, it also presents challenges related to data privacy, security, and intellectual property protection [4]. Organizations hesitate to share their proprietary models due to concerns about model privacy and the potential loss of control over individual models, which can hinder collaboration and innovation [7], [8]. For example, grid models used by Transmission System Operators and Distribution System Operators represent critical infrastructure and contain sensitive information about company assets, which organizations may be unwilling to disclose. Additionally, models and simulation environments may require specialized hardware, specific in-

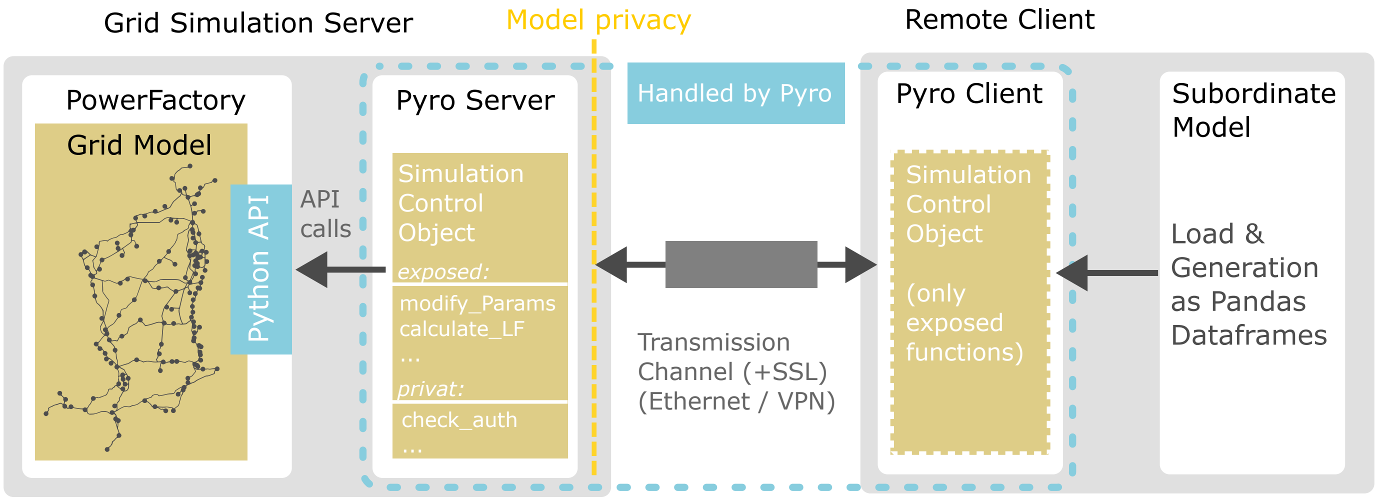

Figure 1: Proposed methodology for distributed power system simulation using Pyro for remote object communication to control and parametrize a PowerFactory grid model.

frastructure, and proprietary licenses. Furthermore, the absence of standardized exchange formats and interfaces, such as the Functional Mock-up Interface (FMI) [9], can further complicate or prevent integration efforts for certain software. Even if such interfaces and data exchange formats are available, highly performant yet also resource-intensive co-simulation frameworks may not be optimal for all use cases, especially those that require only loose couplings, where a simpler and less complicated approach would suffice.

B. Contribution

The present paper introduces a simple, flexible, and secure approach to simulation coupling of electrical power system models using remote object communication to create distributed simulations as illustrated in Figure 1. It leverages Pyro (Python Remote Objects) [10], a Python library originally designed for building distributed applications. For this, we apply Pyro to enable simulation coupling by facilitating remote interaction through predefined objects while sharing only essential information. The resulting distributed simulation ensures secure collaboration and seamless integration across various simulation environments, as it can be applied to any software that provides an Application Programming Interface (API) or, more generally, can be controlled externally by a local program.

The proposed method is demonstrated in power system modeling using DIgSILENT PowerFactory [11] as a proof-of-concept implementation. PowerFactory provides licensable FMI interfaces for RMS and EMT simulations but lacks support for standardized interfaces in load flow calculations. However, it offers a versatile API that provides control over nearly all program functions, which we build upon to develop a distributed simulation using our proposed methodology.

The remainder of this paper is organized as follows: section II reviews related work, while section III outlines the proposed methodology. The section IV presents and discusses the results, and finally, section V summarizes our findings and suggests directions for future work.

II. Related Work

Robust integration mechanisms are essential for enabling simulation coupling across diverse environments in power system modeling. Researchers have explored various approaches to address these challenges, including implementing secure data exchange protocols and developing dedicated co-simulation frameworks.

Co-simulation frameworks have become indispensable tools in multi-domain energy system simulations, allowing heterogeneous models to interact and analyze complex system dynamics [8]. MOSAIK [12] facilitates seamless integration of diverse simulation models and supports large-scale scenario management. Another framework, eASiMOV, provides modular solutions for power grid modeling and analysis, incorporating tools for simulation, visualization, and data management [13]. Both frameworks support interoperability with DIgSILENT PowerFactory, enabling advanced power system simulations. The PROcess Operation Framework (PROOF) [14] focuses on automating computational workflows and modular couplings for system simulations. It leverages process orchestration and message-oriented middleware to streamline co-simulation execution, enhancing automation and scalability. While effective, these frameworks continue to evolve, particularly in enhancing real-time simulation capabilities and improving data exchange efficiency.

Similarly, HELICS, a co-simulation framework for scalable multi-domain modeling [15], excels in large-scale simulations but prioritizes performance over privacy, leaving gaps in secure data exchange. Additionally, its complex setup limits usability for smaller, less resource-intensive applications.

The REFLEX project [16] integrates various bottom-up models for energy supply (electricity, heat, hydrogen) and demand-side sectors (industry, transport, residential). It enables a detailed and dynamic assessment of Europe's transition to a low-carbon energy system by iteratively evaluating energy prices, demand, and system investments.

Several studies have proposed multi-domain coupling models for integrated energy systems [17]–[19]. However, these models are often research-specific, limiting their adaptability across different simulation environments and posing challenges for broader integration.

Despite these advancements, existing approaches tend to be either overly complex, dependent on resource-intensive co-simulation platforms, or narrowly focused on specific tools. This highlights a critical gap in the field: the need for simpler, more flexible methods that facilitate seamless integration across diverse simulation environments while ensuring the protection of sensitive model data.

III. Methodology

This section describes the methodology used to create a distributed simulation using remote object communication on the example of PowerFactory [11] for power grid simulations. Powerfactory, a widely used power system simulation tool, is selected due to its powerful scripting capabilities via a built-in Python API, allowing full simulation control from a local program. The process is structured into three key steps:

1) Step 1: Model Preparation in the Simulation Environment and Parameter Definition: Before initiating the simulation, the accessible parameters from the model are pre-shared with the remote user (client). For example, this can be done by sharing an Excel sheet containing exemplary data. These include both input and result variables, along with possible simulation scenarios.

2) Step 2: Hosting of Simulation Environment: The second step involves hosting the simulation environment (PowerFactory) using Pyro, a Python library that enables remote object communication [10]. PowerFactory is encapsulated within a Python object for simulation control that exposes key functionalities such as running power flow simulations, performing dynamic simulations, and modifying variables of grid elements (e.g., active and reactive power of loads and generation units) while maintaining the privacy of the underlying model. These functions invoke underlying API calls and are registered for remote access using the @expose decorator, as shown in Listing 1.

```
from Pyro5.api import expose

class MyRemoteObject:
    @expose
    def greet(self, name):
        return f"Hello,{name}!"

#create a Pyro daemon
daemon = Pyro5.api.Daemon()
#register the remote object
uri = daemon.register(MyRemoteObject)
#start and keep the server running
daemon.requestLoop()
```

Listing 1: Example - Pyro server exposing a Python function using the @expose decorator.

The remote user connects to the server using the assigned Universal Resource Identifier (URI) and thus gains access to the remote object functions defined using the @expose decorator. To secure communication, Pyro offers a 2-way Secure Sockets Layer (SSL) certificate authentication (Handshake protocol) [20]. This ensures mutual authentication that allows access to be removed by revoking the client certificate. At the same time, no additional information beyond the predefined parameters can be extracted from the model by the remote user.

3) Step 3: Remote Simulation Execution and Result Exchange: In this step, a remote client interacts with the model hosted on the server through the exposed object functions. This allows the client to modify values such as load demand, generation capacity, or other grid parameters, depending on the desired simulation scenario. For example, they might initiate power flow simulations to assess the grid's performance or opt for a quasi-dynamic simulation to study the behavior over time in response to changing conditions.

Pyro handles all the data transfer between the client and server, which is serialized for efficient transmission. However, Pyro's default serializers (Serpent and Marshal) can only handle objects from the Python Standard Library. For commonly used third-party libraries such as Pandas DataFrames and NumPy arrays [22], [23], a custom serializer for DataFrames can be implemented. The registration of the custom serializer in Pyro is shown in Listing 2.

```
import pandas as pd
from Pyro5.api import register_dict_to_class,
    register_class_to_dict

#pandas to a dictionary for serialization
def series_to_dict(obj):
    df=obj.reset_index()
    return { "__class__":"pandas-series","data
        ": df.to_json(orient='split')}

#dictionary back to pandas for deserialization
def series_to_class(classname, d):
    df=pd.read_json(d["data"],orient='split')
    df.set_index(list(df.columns[:-1]),inplace
        =True)
    return pd.Series(df[df.columns[-1]].values
        ,index=pd.MultiIndex.from_tuples(df.
        index.values))

#register the custom serializer to Pyro
register_class_to_dict(pd.Series,
    series_to_dict)
register_dict_to_class("pandas-series",
    series_to_class)
```

Listing 2: Custom serializer for pandas in Pyro5.

IV. Results and Discussion

This section presents the results and discusses Pyro's performance in data transfer and when coupled with the power grid simulation environments. Initial tests were conducted with both the server and client connected via a 1 Gbps local ethernet link. The tests were repeated with the client connected

Data Type	Server/Client	Serializer	Without SSL Speed (MB/s)	With SSL Speed (MB/s)
String	Local Ethernet	Serpent	45.32	45.05
		Marshal	85.50	80.57
DataFrame	Local Ethernet	Serpent	14.46	13.81
		Marshal	18.64	17.64
String	Local Ethernet/VPN	Serpent	2.46	1.78
		Marshal	2.29	2.27
DataFrame	Local Ethernet/VPN	Serpent	1.91	1.73
		Marshal	1.88	1.85

Table 1: Pyro data transfer performance for different serializers, SSL encryption, and network conditions.

via a 100 Mbps coaxial cable internet connection through a Virtual Private Network (VPN). In both scenarios (local ethernet/VPN), standard Windows 11 PCs with Python 3.11 were used, with Pyro version 5 handling the client-server communication.

A. Pyro Communication Performance

To evaluate Pyro's efficiency, we test raw data transfer speeds, the impact of different serialization methods, the effect of SSL encryption and authentication, and behavior under reduced internet speed. The results are summarized in Table 1. In the local ethernet connection, both the Serpent and Marshal serializers perform efficiently, achieving values close to those stated in the Pyro documentation. The Marshal serializer achieves the highest transfer speeds, reaching 85.50 MB/s for strings, while the Serpent serializer records 45.32 MB/s for strings. However, when transmitting DataFrames, transfer speeds drop significantly for both serializers (Marshal at 18.64 MB/s and Serpent at 14.46 MB/s). In a VPN connection, the difference in data transfer speed between DataFrames and strings is less than 25% and is not much affected by the serializer. This is because, by default, Pyro serializers cannot handle complex data structures like DataFrames directly (for the test, we converted the DataFrame to JSON before serializing).

In a local ethernet connection, Marshal outperforms Serpent due to its simplified binary format and more direct approach. While Serpent's additional computational steps (like UTF-8 encoding) introduce higher overhead. However, in a VPN connection, where bandwidth limitations dominate, serialization overhead becomes less significant relative to network-induced latency.

SSL encryption has a minimal impact on transfer speeds in a local ethernet connection, reducing them by less than 6% for both serializers. However, in a VPN connection, SSL significantly affects Serpent more than Marshal, reducing Serpent's transfer speed by up to 27%, while Marshal remains largely unaffected. The higher bandwidth of local ethernet easily compensates for the additional delay introduced by SSL, offsetting encryption time and latency. In contrast, in a VPN with limited bandwidth, SSL's relative overhead becomes noticeable, further adding to the existing transmission latencies.

B. Evaluation of Pyro Performance with Simulation Environments

The evaluation of Pyro's integration with simulation environments (PowerFactory) utilized two distinct grid models: the D76 110 kV distribution grid (DSO) and a large-scale German transmission grid (TSO). The DSO model, representing the D76 control zone in Germany, consists of 140 substations, 300 transformers, and 140 synchronous machines and is illustrated in Figure 2. In contrast, the TSO model is significantly larger, containing 745 substations, 945 transformers, and 489 synchronous machines.

The execution times of the test cases are summarized in Table 2. The activation time, which includes the client-server connection, SSL authentication (if enabled), access to PowerFactory, and model initialization, remains consistently low across all test cases. This is primarily due to Pyro's efficient handling of the initial connection, which minimizes overhead. Among activation time, model initialization is the most time-consuming step. While this step is independent of Pyro, it is heavily influenced by the performance of the simulation tool, hardware configuration, and the complexity of the initialized grid model.

The most time-consuming phase across all test cases is the parameter definition time, which includes defining, serializing, transmitting, and modifying model parameters on the server. In the larger TSO model, modifying 10 load units and querying 20 generator outputs taken 1.73 seconds, while modifying 233 load units and querying 20 generator outputs in the smaller DSO model takes 3.51 seconds. The increase in time is due to the greater number of parameters shared, which requires more data to be serialized, transmitted, and updated in the model, and is independent of the size or complexity of the model. SSL encryption and authentication have a minimal effect on these processes, with the difference between scenarios with and without SSL being less than 2%. This indicates that the encryption overhead does not cause significant delays when compared to the time spent on the parameter definition phase (these effects are described isolated in subsection IV-A).

Grid	SSL	Client/Server	Activation (s)	Para Def. (s)	Sim Exec. (s)	Total (s)
DSO	No	Local Ethernet	0.01	3.51	0.08	3.61
	Yes	Local Ethernet	0.07	3.50	0.08	3.65
	No	Local Ethernet/VPN	0.09	3.91	0.12	4.12
	Yes	Local Ethernet/VPN	0.20	3.93	0.12	4.25
TSO	No	Local Ethernet	0.01	1.73	0.89	2.63
	Yes	Local Ethernet	0.07	1.74	0.90	2.71
	No	Local Ethernet/VPN	0.09	1.84	0.96	2.89
	Yes	Local Ethernet/VPN	0.21	1.87	0.97	3.05

Table 2: Performance evaluation of Pyro integration with PowerFactory.

The method successfully manages both the DSO grids and TSO grids and executes power flow simulation in 3.61s and 2.63s, respectively, accommodating models of varying complexities. Once a remote connection is established, the scalability is unaffected by model complexity, as it is handled by the system executing the simulation (i.e., the server machine). However, parameter exchange is influenced by factors such as serialization methods and internet speed, depending on the size and structure of the transmitted data. The methodology ensures that grid models are securely stored on the server, exposing only predefined parameters to the client, thereby maintaining model privacy. To further enhance security, the model server can only be accessed through a VPN, allowing

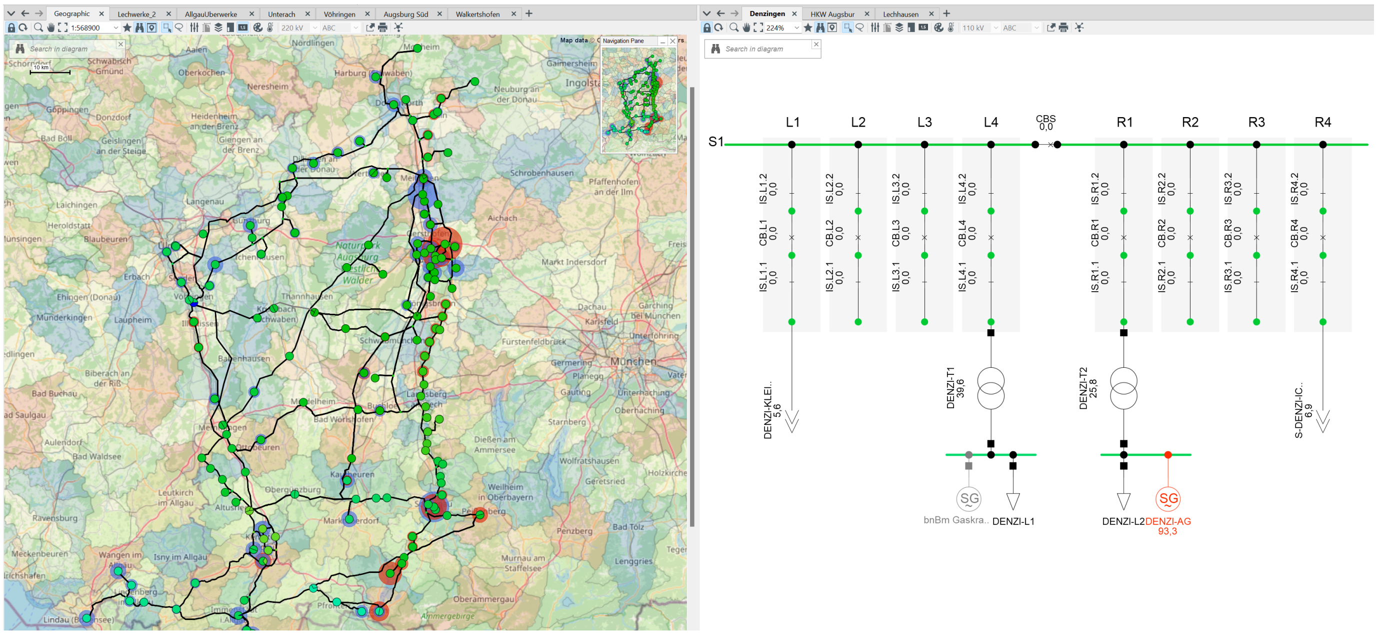

Figure 2: PowerFactory model of control zone D76 of Germany with load and generation remotely parameterizable through Pyro. Left: A geographic display of the calculated power flow, with the resulting load and generation illustrated as circles. Right: A detailed schematic diagram of a substation showing aggregated load and generator elements.

only authorized clients or users to connect. Third-party access is strictly prohibited, and communication is secured through SSL certificate authentication, ensuring that only clients with valid certificates can interact with the system. Access rights can also be easily revoked, providing full control over who can access the models at any time.

C. *Evaluation of Pyro with other Simulation Coupling Frameworks*

Pyro offers a lightweight solution for loosely coupled simulations due to its minimal setup requirements. However, it lacks built-in support for co-simulation features such as orchestrators and scenario configurations, which limits its applicability in tightly coupled simulations [21]. In contrast, HELICS [15] and MOSAIK [12] are more specialized frameworks with such capabilities, making them better suited for tightly coupled simulations that demand high performance, high data throughput, and low latency.

Directly comparing the performance of these frameworks is challenging, as latency and data throughput are influenced by various factors, including hardware architecture and network conditions. Ultimately, the choice of a coupling framework depends on factors such as the type of simulation models, timing requirements, and language interoperability. While Pyro may not be ideal for high-fidelity co-simulation, its simplicity and flexibility allow users to manually implement the necessary co-simulation features when needed, making it a practical, low-overhead solution for simpler simulation coupling.

V. Conclusion and Future Works

Simulation coupling using Pyro for distributed power system models demonstrates notable flexibility and ease of use, facilitating efficient model coupling and smooth data exchange. A major advantage of Pyro is its ability to handle remote objects as if they were local, which significantly simplifies the development of simulation coupling frameworks by eliminating the need for extensive low-level networking or Marshaling code. In theory, even more complex data structures, such as simulation submodules encapsulated in Python objects, can be shared through Pyro. These results demonstrate that the proposed approach is highly effective for simulation coupling, providing an optimal balance of simplicity, flexibility, security, and performance. By leveraging Pyro's capabilities in building distributed systems with minimal programming effort, various simulation environments with direct API access can be flexibly coupled. This approach effectively addresses critical challenges in simulation coupling, such as ensuring model privacy, scalability, and efficient data exchange, offering a practical and adaptable solution for collaborative research where multiple participants can couple models without compromising confidentiality. Our benchmarking tests show that this flexibility comes at the cost of increased data exchange times compared to theoretical limits. However, the impact of this time increase is negligible for loosely coupled simulations, where the simulation time defines the overall performance. The proposed simulation coupling methodology via remote object communication can also be implemented using frameworks like Pyrolite for Java and .NET, and can integrate with any simulation tool with API access.

Future work will focus on extending the approach by coupling it with widely used simulation tools such as Pandapower [22] and PyPSA [24]. Additionally, models for electricity markets, weather data, cyber-physical power systems, and multi-energy systems—including gas and hydrogen interconnected with power grids to be coupled to enable comprehensive, holistic, and cross-domain simulations in power systems.

VI. Acknowledgements

This work is funded by the Helmholtz Platform for the Design of Robust Energy Systems and their Supply Chains (RESUR).

References

[1] M. Chang, H. Lund, J. Zinck Thellufsen, and P. A. Østergaard, "Perspectives on purpose-driven coupling of energy system models," *Energy*, vol. 265, p. 126335, 2023. [Online]. Available: https://doi.org/10.1016/j.energy.2022.126335.

[2] T. Luan, "A Comprehensive Review of Simulation Technology: Development, Methods, Applications, Challenges, and Future Trends," in International Journal of Emerging Technologies and Advanced Applications, vol. 1, pp. 9-14, Jun. 2024. Available: https://doi.org/10.62677/IJETAA.2405119.

[3] B. Wiegel, T. Steffen, D. Babazadeh, and C. Becker, "Towards a more comprehensive open-source model for interdisciplinary smart integrated energy systems," in 2023 11th Workshop on Modelling and Simulation of Cyber-Physical Energy Systems (MSCPES), pp. 1-7, 2023. Available: https://doi.org/10.1109/MSCPES58582.2023.10123432.

[4] M. Fodstad, P. Crespo del Granado, L. Hellemo, B. R. Knudsen, P. Pisciella, A. Silvast, C. Bordin, S. Schmidt, and J. Straus, "Next frontiers in energy system modelling: A review on challenges and the state of the art," in Renewable and Sustainable Energy Reviews, vol. 160, p. 112246, 2022. Available: https://doi.org/10.1016/j.rser.2022.112246.

[5] T. Kuhn, P. O. Antonino, and A. Bachorek, "A simulator coupling architecture for the creation of digital twins," *Publica Fraunhofer*, 2020. [Online]. Available: https://publica.fraunhofer.de/handle/publica/409727. [Accessed: 12-Feb-2025].

[6] J. Jiang, H.-L. Shen, Y. Xia, and H. H. C. Iu, "Optimal coupling pattern of cyber-physical systems," in *Proc. 2021 IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2021, pp. 1-5. Available: https://doi.org/10.1109/ISCAS51556.2021.9401252.

[7] H. Yang, Y. Bai, Z. Zou, Y. Shi, and R. Yang, "Research on the Security Sharing Model of Power Grid Data Based on Federated Learning," in 2021 IEEE 5th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), vol. 5, pp. 1566-1569, 2021. Available: https://doi.org/10.1109/ITNEC52019.2021.9587301.

[8] M. Vogt, F. Marten, and M. Braun, "A survey and statistical analysis of smart grid co-simulations," *Applied Energy*, vol. 222, pp. 67-78, 2018. Available: https://doi.org/10.1016/j.apenergy.2018.03.123.

[9] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, and others, "The functional mockup interface for tool independent exchange of simulation models," in *Proc. 8th Int. Modelica Conf.*, Linköping University Press, 2011, pp. https://doi.org/10.3384/ecp11063105.

[10] I. de Jong, "Pyro5 Documentation." Available: https://pyro5.readthedocs.io, Accessed: Jan. 2025.

[11] DIgSILENT, PowerFactory 2023 User Manual, 2023. DIgSILENT GmbH, *PowerFactory*, Available: https://www.digsilent.de/.

[12] C. Steinbrink, M. Blank-Babazadeh, A. El-Ama, S. Holly, B. Lüers, M. Nebel-Wenner, R. P. Ramírez Acosta, T. Raub, J. S. Schwarz, S. Stark, A. Nieße, and S. Lehnhoff, "CPES Testing with mosaik: Co-Simulation Planning, Execution and Analysis," *Applied Sciences*, vol. 9, no. 5, article 923, 2019. Available: https://doi.org/10.3390/app9050923.

[13] H. Çakmak, M. Kyesswa, U. Kühnapfel, and V. Hagenmeyer, "eASiMOV - A New Framework for Power Grid Analysis," Jun. 2018, Available: https://doi.org/10.13140/RG.2.2.15952.20480.

[14] J. Liu, M. Chlosta, N. Schaber, J. Sidler, R. Lutz, T. Schlachter, and V. Hagenmeyer, "Introducing PROOF - A PROcess Orchestration Framework for the Automation of Computational Scientific Workflows and Co-Simulations," in 2023 Open Source Modelling and Simulation of Energy Systems (OSMSES), pp. 1–6, Aachen, Germany, Mar. 2023. Available: https://doi.org/10.1109/OSMSES58477.2023.10089680.

[15] 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," in IEEE Access, vol. 12, pp. 24325–24347, 2024. Available: https://doi.org/10.1109/ACCESS.2024.3363615.

[16] Kunze, R., & Schreiber, S. (2021). Model Coupling Approach for the Analysis of the Future European Energy System. In D. Möst, S. Schreiber, A. Herbst, M. Jakob, A. Martino, & W.-R. Poganietz (Eds.), *The Future European Energy System: Renewable Energy, Flexibility Options and Technological Progress* (pp. 27–51). Springer International Publishing. Available: https://doi.org/10.1007/978-3-030-60914-6_3.

[17] T. Kang, K. Liu, M. Bai, M. Wu, D. Jia, and E. Luo, "Research on Modeling and Joint Power Flow Simulation of Multi Energy Coupling Network," in Proc. 2022 5th Int. Conf. Energy, Electrical and Power Engineering (CEEPE), pp. 1165-1170, 2022. Available: https://doi.org/10.1109/CEEPE55110.2022.9783293.

[18] P. Sun, S. Wu, L. Chen, M. Yao, S. Zhang, and X. Zhang, "Gas-Electric Power Coupling Simulation Method for Integrated Energy System Based on Improved MCMC," in Proc. 2023 5th Int. Conf. Power and Energy Technology (ICPET), pp. 1569-1575, 2023. Available: https://doi.org/10.1109/ICPET59380.2023.10367559.

[19] X. Tian, L. Zhouhong, P. Zhaoguang, and H. Sun, "Modeling and Simulation for Multi Energy Flow Coupled Network Computing," in Proc. 2018 Int. Conf. Power System Technology (POWERCON), pp. 992-998, 2018. Available: https://doi.org/10.1109/POWERCON.2018.8602267. .

[20] A. O. Freier, P. Karlton, and P. C. Kocher, "The Secure Sockets Layer (SSL) Protocol Version 3.0," *RFC 6101*, RFC Editor, Aug. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6101. [Accessed: Feb. 6, 2025].

[21] L. Barbierato, P. Rando Mazzarino, M. Montarolo, et al., "A comparison study of co-simulation frameworks for multi-energy systems: the scalability problem," *Energy Informatics*, vol. 5, Suppl. 4, p. 53, 2022. Available: https://doi.org/10.1186/s42162-022-00231-6.

[22] L. Thurner et al., "pandapower — An Open-Source Python Tool for Convenient Modeling, Analysis, and Optimization of Electric Power Systems," in *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 6510-6521, Nov. 2018, Available: https://doi.org/10.1109/TPWRS.2018.2829021.

[23] D. Lohmeier, D. Cronbach, S. R. Drauz, M. Braun, and T. M. Kneiske, "Pandapipes: An open-source piping grid calculation package for multi-energy grid simulations," *Sustainability*, vol. 12, no. 23, Art. no. 9899, 2020. [Online]. Available: https://www.mdpi.com/2071-1050/12/23/9899.

[24] T. Brown, J. Hörsch, and D. Schlachtberger, "PyPSA: Python for Power System Analysis," in *Journal of Open Research Software*, vol. 6, no. 1, article 4, 2018. Available: https://doi.org/10.5334/jors.188.