PNNL-38935

Multi-Entity Simulation with CoSim Toolbox

December 2025

Trevor D Hardy
Mitch A Pelton
Nathan T Gray
Fred C Rutz
Bo Wen

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor Battelle Memorial Institute, nor any of their employees, makes **any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights**. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or Battelle Memorial Institute. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from
the Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062
www.osti.gov
ph: (865) 576-8401
fox: (865) 576-5728
email: reports@osti.gov

Available to the public from the National Technical Information Service
5301 Shawnee Rd., Alexandria, VA 22312
ph: (800) 553-NTIS (6847)
or (703) 605-6000
email: info@ntis.gov
Online ordering: http://www.ntis.gov

PNNL-38935

Multi-Entity Simulation with CoSim Toolbox

December 2025

Trevor D Hardy
Mitch A Pelton
Nathan T Gray
Fred C Rutz
Bo Wen

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99354

Summary

Co-simulation is an analysis technique for linking multiple software models during runtime by facilitating data exchange and simulation time synchronization. There are numerous challenges when constructing an effective co-simulation including simulation tool installation, data management, and writing new models in a manner compatible with the co-simulation framework of choice. CoSim Toolbox is an integration of multiple pieces of software designed to make assembling such a co-simulation in HELICS easier. This report summarizes the existing capabilities of CoSim Toolbox and outlines future development plans.

Acknowledgments

This research was supported by the Energy System Co-Design with Multiple-Objectives and Power Electronics (E-COMP) Initiative under the Laboratory Directed Research and Development (LDRD) Program at Pacific Northwest National Laboratory (PNNL). PNNL is a multi-program national laboratory operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL01830.

Acronyms and Abbreviations

FMI	Functional Mockup Interface
HLA	High-Level Architecture
CST	CoSim Toolbox

Contents

1.0 Introduction

Co-simulation is a technique that merges multiple models running in (generally) heterogenous software to create a larger, more complex model that enables analysis that would not otherwise be possible. These models are linked through a co-simulation platform that enables data exchange during simulation runtime. Typically, the data exchanges that take place are boundary conditions for the individual models that would, when not running in a co-simulation, be satisfied through static or historical values. Co-simulation, by enabling dynamic data exchange allows for system models that transcend individual simulation tools, modeled domains, and modeled subsystems.

There are a number of common co-simulation platforms or technologies such as FMI (Modelica Association Project FMI 2025), HLA (IEEE 2025), Mosaik (Steinbrink, et al. 2019) and HELICS (Hardy, et al. 2024). All of these platforms provide a generic means of linking models, with details of said models abstracted away from the co-simulation platform itself, and various other features for building and running a co-simulation. It is also common for modelers to create a one-off co-simulation implementation that is able to link two specific models needed for a particular analysis. Such implementations are not general and thus would not be considered a co-simulation platform.

HELICS is the US Department of Energy's publicly available co-simulation platform and has been under development since 2017. HELICS provides both a means of producing the data exchanges necessary to link models and also the time-management algorithm to keep individual models appropriately synchronized in time. HELICS has been used in a large number of studies covering topics such as the evaluation of transactive retail power markets (Theisen, et al. 2024), the impact of inverters on power system transients (Bharati, et al. 2023), the impact of the charging of electric vehicles in a city (Panossian, et al. 2023), and cyber security analysis of power system operations (Lardier 2020).

Through the years of developing and using HELICS, it has become apparent that there are several challenges when building and running a HELICS-based co-simulation. Broadly, these fall into a few categories:

- Simulation tool installation – Each model may use its own simulation tool and thus require installing said tool on a given computer to allow the co-simulation to run.

- Configuration management – Defining how each tool is to be run and the data exchanges to be made is often laborious and must be created for each particular co-simulation being run.

- Data management – Each simulation tool being used produces data in its own format and stores it on disk in its own way (*e.g.* file, database). Accessing these results may require a significant amount of custom data management code.

- Model development – In situations where a new custom model needs to be developed, users may be required to learn more of the underlying simulation platform then they would otherwise like; this can be time-consuming and may prevent users from even undertaking the construction of the co-simulation.

To begin to address these challenges, CoSim Toolbox (CST) was developed (CoSim Toolbox Github n.d.). CST is largely an integration of existing tools along with custom Python classes that are intended to reduce the burden of building and running a co-simulation. The remainder of this report will document CST's features as they address the above challenges.

2.0 Challenge 1: Simulation Tool Installation

Simulation tools across analysis domains target a variety of users and assume a variety of installation environments. To address this as much as possible, CST uses Docker (Docker n.d.) as the primary means of distributing the integrated co-simulation environment. Containerization provided by Docker is a technology that provides abstraction and isolation from the underlying computation environment and thus provides a consistent environment in which the simulation tool can run. Thus, independent of the computer on which it is installed, any Docker-ized simulation tool can be expected to run.

CST itself provides a Docker container with the supporting tools it uses (see below) as well as the CST Python API (also see below). Additionally, a few simulation tools have been installed in CST-enabled Docker containers, allowing them to be downloaded and easily installed. The expectation is that additional tools will be adapted to make native CST versions in the coming years and a collection commonly used CST-ready simulation tools will be available.

It is worth noting that there is a software development cultural split in the power systems community. Many of the tools used in research are developed to run on all of the major operating system platforms (macOS, Linux, and Windows) while most of the industry tools are only available on Windows. Such Windows-only tools are not, at this time, able to be Docker-ized and thus are not able to be installed via Docker. These tools are, generally, still able to be used in a CST co-simulation, they only need to be manually installed and managed.

3.0 Challenge 2: Configuration Management

Configuring the time management and data exchange parameters for all models in a co-simulation is a detailed and generally time-intensive task. In HELICS this configuration often takes the form of a per-model JSON file on disk that the model references when it joins the co-simulation. For large co-simulations with many models, a large number of these configuration files accumulate on the computer that is running the co-simulation. Given this proliferation, it is often not always clear which files were used by which models or for which analysis scenarios.

Additionally, there can be scenario-specific configuration or more general metadata that is used either by the models or by other code (*e.g.* model creation, pre- or post-processing data). Access to this configuration data is essential for the correct understanding and interpretation of the simulation results.

To help users manage this metadata, CST provides a centralized database to hold this kind of structured data. CST uses a Mongo database (MongoDB n.d.) to hold both model-specific configuration data as well as arbitrary user-provided data. This database is distributed as part of the CST Docker and is instantiated on launch. This database provides a persistent store for data and allows users across the analysis team to access configuration and other metadata for use. By centralizing this data, confusion about the configuration for a given scenario can be avoided.

When using the CST-provided class for a given model (see below), the model can be configured to query the database for its co-simulation configuration information automatically, preventing the need for any such data to be stored locally on disk. APIs for accessing the database have been created as a part of the CST API and allow users with no knowledge of Mongo to write and read data from the database. The database itself can be manually inspected through the use of the included mongo-express web service (mongo-express n.d.).

CST also provides the capability of using the local disk as a repository for this data instead of the Mongo database. Local disk writes provide higher performance at the cost of providing the analysis team easy access to this data.

4.0 Challenge 3: Data Management

Like the challenges around configuration and metadata, there are similar challenges to managing the model input and output data itself. These data files may be large, may be of a diverse format, and may reside on a remote computer system. All of these present barriers for users trying to get data into and out of their co-simulation model. To reduce these barriers, CST provides Postgres (PostgreSQL n.d.) as a general time-series database for use by the co-simulation model. This database can be used to store time-series data models may need during the simulation as well as holding output data produced by the models.

CST has created a Logger application that collects all data that is transmitted between models during the co-simulation and writes it into the time-series database. This automated data collection allows users to log any data for later analysis simply by producing it as an output from a given model; it is not required that any other model receive it as an input for it to be logged. As in the situation of storing configuration and metadata in Mongo, APIs are provided for writing and reading data into and from the time-series database. This provides a means for an analysis team to both prepare for a co-simulation by pushing data into the time-series database as well as analyze the post-simulation results. The pgAdmin web service (pgAdmin n.d.) is also included to allow for manual inspection of the database.

CST also provides a means of using the local disk as the data store for time-series data. Local disk writes provide higher performance at the cost of providing the analysis team easy access to this data.

5.0 Challenge 4: Model Development

For many users seeking to execute a co-simulation, the use of the underlying co-simulation platform presents a significant barrier. It always involves learning a new tool or set of APIs and this can be significant investment. CST seeks to reduce this barrier by providing a generic class for implementing models where the HELICS APIs are wrapped to provide a more simplified and generalized interface. Almost all of the implemented methods have reasonable defaults such that, in many cases, the user does not need to specialize the functionality and can use the code as-is. This dramatically simplifies the task of implementing a new model and allows the users to focus on data exchanges and the implementation of the model proper.

The CST class also includes the necessary methods for using the various data management backends to read and write configuration and time-series data, be that the centralized, database data stores (MongoDB and PostgreSQL) or the local data stores (JSON and CSV). When creating a new model, the user defines which data stores are being used and the class takes care of the rest. The data handling functionality is written in a modular way that allows support for new data stores to be easily added. For example, there is interest in supporting several alternative local data store formats (*e.g.* parquet and HDF5) and databases (*e.g.* time-series data storage in MongoDB) and the software architecture the CST API allows these data stores to be added in relatively easily.

6.0 Current and Future State of CST

After working with the Commercialization Office, and given their consideration, it was determined that CST should be released as open-source software. CST has had an initial release on Github with full documentation and modest amounts of automated testing. In working with existing HELICS users and presenting CST to them, there has been agreement on the challenges of building co-simulations and appreciation for the goals and the improvements that CST provides. Several existing projects show interest in adopting CST, not the least of which is the integrated analysis task in E-COMP where CST laid the foundation for the development of PySO, a generic model for bulk power system markets.

The feature set for CST is far from complete; development is anticipated in the following areas:

- **Monitoring and debugging support** – Co-simulation users need the equivalent of an integrated development environment for monitoring and debugging their co-simulations. Currently this debugging takes place through the examination of multiple log files, printing statements to console and debugging a single model by stepping through its code. Given the integration of multiple models, tooling that is able to manage these models in a single environment with the ability to pause the co-simulation and evaluate the data exchanges between them would provide significant value when developing the co-simulation. Currently CST has beta functionality using Grafana, a web-based graphing tool, to query the time-series database and show the existing collected data but much greater functionality is needed.

- **Co-simulation profiling** – It is not unusual for co-simulation performance to be a problem and tooling to collect appropriate data to find the cause of co-simulation slow-downs would be valuable. Collecting this data is not challenging but correctly attributing it such that the culprit model can be identified is non-trivial.

- **Federation data-exchange definition** – A consistent pain point for co-simulation users is creating the definitions for the data exchanges between models. Particularly once the number of data exchanges (either within a single federate or across a large number of federates) grows sufficiently, defining the data exchanges must be done programmatically. Developing tooling that allows users to do this more quickly and with more confidence would remove a consistently unpleasant part of building a co-simulation.

7.0 References

Bharati, Alok Kumar, Venkataramana Ajjarapu, Wei Du, and Yuan Liu. 2023. "Role of Distributed Inverter-Based-Resources in Bulk Grid Primary Frequency Response Through HELICS Based SMTD Co-Simulation." *IEEE Systems Journal* 1071-1082.

n.d. *CoSim Toolbox Github.* https://github.com/pnnl/cst.

Docker. n.d. Accessed 12 2025. https://www.docker.com/.

Hardy, Trevor D, Bryan Palminteir, Phillip Top, Dheepak Krishnamurthy, and Jason Fuller. 2024. "HELICS: A Co-Simulation Framework for Scalable Multi-Domain Modeling and Analysis." *IEEE Access.*

n.d. *https://github.com/mongo-express/mongo-express.* Accessed 12 2025.

IEEE. 2025. "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)--Framework and Rules." *IEEE Std 1516-2025 (Revision of IEEE Std 1516-2010)* (IEEE) 1-40.

Lardier, William. 2020. *ASGARDS-H: Enabling Advanced Smart Grid Cyber-Physical Attacks, Risk and Data Studies with HELICS.* Montreal: Concordia University.

Modelica Association Project FMI. 2025. *FMI Github.* Accessed 11 24, 2025. https://github.com/modelica/fmi-standard.

MongoDB. n.d. Accessed 12 2025. https://www.mongodb.com/.

mongo-express. n.d. Accessed 12 2025. https://github.com/mongo-express/mongo-express.

Panossian, Nadia V, Haitam Laarabi, Keith Moffat, Heather Chang, Bryan Palmintier, Andrew Meintz, Timothy E Lipman, and Rashid A Waraich. 2023. "Architecture for Co-Simulation of Transportation and Distribution Systems with Electric Vehicle Charging at Scale in the San Francisco Bay Area." *Energies.*

pgAdmin. n.d. Accessed 12 2025. https://www.postgresql.org/.

PostgreSQL. n.d. Accessed 12 2025. https://www.postgresql.org/.

Steinbrink, C, M Blank-Babazadeh, A El-Ama, S Holly, B Lüers, M Nebel-Wenner, R P Ramirez Acosta, et al. 2019. "CPES Testing with mosaik: Co-Simulation Planning, Execution and Analysis." *Applied Sciences.*

Theisen, John R, Bose Anjan, Monish Mukherjee, Dan Burgess, Kenneth Wilhelm, and Michael Diedesch. 2024. "Community-Based Transactive Coordination Mechanism for Enabling Grid-Edge Systems." *2024 IEEE Texas Power and Energy Conference (TPEC).* College Station: IEEE. 1-6.

Pacific Northwest National Laboratory

902 Battelle Boulevard
P.O. Box 999
Richland, WA 99354

1-888-375-PNNL (7665)

www.pnnl.gov