INL/CON-22-70071-Revision-1

Digital Twin for Optimizing Real-time Economy of the Integrated Energy Systems

July 2023

Changing the World's Energy Future

Takanori Kajihara, Daniel Garrett, Junyung Kim, Linyu Lin, Paul W Talbot, Jeren M Browning

Idaho National Laboratory

INL/CON-22-70071-Revision-1

Digital Twin for Optimizing Real-time Economy of the Integrated Energy Systems

Takanori Kajihara, Daniel Garrett, Junyung Kim, Linyu Lin, Paul W Talbot, Jeren M Browning

July 2023

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

Digital Twins for Optimizing the Real-Time Economy of Integrated Energy Systems

Takanori Kajihara[1], Daniel Garrett[1], Junyung Kim[1], Linyu Lin[1], Jeren M. Browning[1], and Paul Talbot[1]

[1]Idaho National Laboratory, Idaho Falls, ID

Takanori.Kajihara@inl.gov

ABSTRACT

Economic and safe operation of integrated energy systems (IES) requires real-time optimization (RTO) of the control and actions conducted on each system component. In this regard, digital twins (DTs), which consist of a physical system, a virtual system, and the data communication that occurs between the two, are essential for effective RTO. Through the data warehouse, the virtual system is constantly updated with real-time data from the physical system, and functions as the model in the optimization framework. The reduced-order model of the dynamic process model in the virtual system is used in the optimization framework. The optimization results are then returned, via the data warehouse, as control actions to the physical system. This work demonstrates the software capabilities of DT assets for an IES in the context of preparing a DT for an experimental system comprised of Idaho National Laboratory (INL)'s Thermal Energy Delivery System and battery system. For the virtual demonstration, the DTs encompass (1) a physical system, including the Modelica models of the Thermal Energy Delivery System and the battery system; (2) virtual optimization via the Optimization of Real-Time Capacity Allocation (ORCA) platform; and (3) the open-source data warehouse software DeepLynx. This work assesses the performance of ORCA, which utilizes a reduced-order model built using the Risk Analysis Virtual Environment (RAVEN) and trained on the Modelica models and real-time data pipeline through the graph database hosted in DeepLynx. The proposed optimization workflow will be an RTO model based on DTs and the data they generate.

Keywords: Real-time optimization, IES, ORCA, DeepLynx, RAVEN

1. INTRODUCTION

Integrated energy systems (IES) are intricate industrial processes that involve multiple operations and components, each with its own goals and interconnections with other parts. To ensure safe and efficient operation of an IES, it must be optimized at various time scales. Typical optimization and control hierarchies can vary from weeks to seconds, depending on the functions. Real-time optimization (RTO) enables economic optimization by reducing operational costs and maximizing revenue, typically in minutes to hours.

RTO is a workflow in which decision variables (i.e., setpoints) are optimized by solving a mathematical optimization problem using a system model and an economic model with process constraints [1]. In the classical method that originated in the late 1980s, a steady-state model depicts the behavior of the given process and optimizes the economical objective function while adhering to the steady-state process model [2]. Among the various RTO approaches are model parameter adjustment [3], integrated system optimization and parameter estimation [4], modifier adaptation [5], and economic model predictive control (EMPC) [6].

The digital twin (DT) [7]—an essential element of RTO—encompasses the physical system, the lifecycle-spanning virtual representation thereof, and the data communication that occurs between the two. The virtual model enables real-time simulation of the behavior and characteristics of the physical system. Once the optimization is performed using the updated computational model, the results are transmitted back to the physical system as recommended optimal setpoints for the physical system's future operation.

This paper demonstrates the software capabilities of DT assets for an IES in the context of preparing a DT for an experimental system comprised of Idaho National Laboratory (INL)'s battery system and Thermal Energy Delivery System. Optimization of energy charging/discharging with the battery system will be demonstrated virtually as a simplified example of RTO. The outcome of the virtual demonstration will be applied to the gap analysis aimed at realizing an actual demonstration connecting to the physical system. The fundamental components of a DT/RTO workflow are shown in Fig. 1.

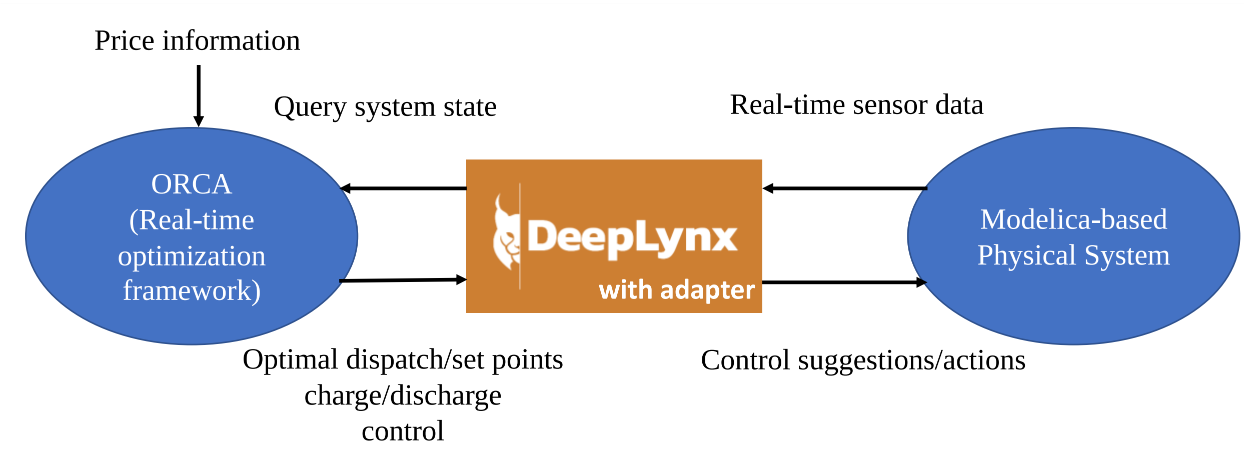

Figure 1. Optimization workflow, carried out through the DeepLynx data warehouse.

The workflow consists of three main components: (1) a physical system, including the Modelica [8] models of the battery system; (2) virtual optimization via the Optimization of Real-Time Capacity Allocation (ORCA) platform [9]; and (3) the open-source data warehouse software DeepLynx [10]. The Modelica-based virtual models of the physical system send real-time sensor data to DeepLynx, which then stores the data in its graph database. The stored data are promptly queried and transferred to ORCA. ORCA determines optimized control parameters by relying on a reduced-order model, then sends, through DeepLynx, the control actions to the Modelica model-based physical system. This work evaluates how well ORCA works when using a reduced-order model created using the Risk Analysis Virtual Environment (RAVEN) [11], based on Modelica models and real-time data sent through the DeepLynx graph database.

ORCA helps analysts create and run real-time control and optimization workflows for DTs. It provides tools that help analysts connect to data sources such as DeepLynx, write their own real-time control and optimization workflows, read stateful data about facility operations, and optimize control to boost profits and resilience. ORCA is especially useful for IES that must respond to external signals and stimuli. DeepLynx allows users to store data in a graph database with a custom ontology. This helps big projects tap into digital engineering and connect with different software systems through DTs. The data are stored as nodes that have relationships with each other. Users query data by utilizing the nodes and relationships. Thus, a more dynamic data model is achieved through the custom ontology, along with a more intuitive approach to querying complex data (compared to traditional methods such as a relational database using Structured Query Language [SQL]).

2. METHODS

2.1. Methods for Real-Time Optimization Workflow Using ORCA

The ORCA workflow is based on a receding-horizon- or EMPC-based optimization [6]. EMPC uses a dynamic model to optimize an objective function over a finite time horizon. The decision variable for the optimization problem is the input trajectory (charging and discharging) over that time horizon. The system (state of charge [SOC]) model serves as an additional constraint to the optimization. The IES model used for this RTO workflow reflects a nuclear power plant (NPP) operating at constant capacity and an electrical storage device that can charge from the NPP and then discharge to the real-time market. The NPP directly sends electricity to either the real-time market or the electrical storage device. Meanwhile, the electrical storage device is characterized by two parameters: (1) power rating (MW), specifying the amount of power the electrical storage device can charge or discharge; and (2) SOC (MWh), specifying the amount of energy being stored by the electrical storage device. This work couples a battery model in Modelica [8] with a NPP that produces a constant electricity output q_{NPP}. As a result, the battery SOC S_t can be described as a function of charging q^C and discharging q^D power rates at time step t:

$$S_t = A\begin{bmatrix} q_{NPP} \\ S_{t-1} \end{bmatrix} + B\begin{bmatrix} q_t^C \\ q_t^D \end{bmatrix}, \tag{1}$$

where A and B are coefficient matrices that depend on the battery properties, including maximum power rating, maximum and minimum SOC, and conversion efficiency. Given a fluctuating electricity price P_t, the revenues can be maximized by finding the optimal charging and discharging power rates:

$$\max \sum_{t=1}^{T} P_t(q_{NPP} - q_t^C + q_t^D), \tag{2}$$

where P_t is the locational marginalized pricing in \$/MW at time step t, and q_{NPP} is the constant power generation of the NPP (MW). The objective function is to search for optimal combinations of charging and discharging power rates that satisfy the following constraints:

$$0 \leq S_t \leq S_{max}, \tag{3}$$

$$0 \leq q_t^C \leq q_{max}^C, \tag{4}$$

$$0 \leq q_t^D \leq q_{max}^D, \tag{5}$$

$$0 \leq q_t^D \leq \sqrt{\gamma_{RTE}} S_t + \gamma_{RTE} q_t^C, \tag{6}$$

where S_{max} is the energy capacity (MWh) of the battery; q_{max}^C and q_{max}^D are the maximum charging and discharging power ratings (MW), respectively, based on the manufacturing specifications; Δt is the user-assigned unit time horizon, and γ_{RTE} is the conversion loss when energy is stored during the charge phase and released during the discharge phase. The present work uses RAVEN to calibrate coefficient matrices A and B, based on transient data from the battery model.

ORCA is solved in a receding horizon fashion. At a given time τ, EMPC receives a state measurement (SOC at τ) used for initializing the dynamic model. An optimal piecewise input trajectory for the objective function and constraints is computed over the time horizon, corresponding to times $t \in [\tau, \tau + \Delta t, \tau + 2\Delta t, \ldots, \tau + N\Delta t]$, where N is the total number of time steps. The first of the optimized input trajectories (charging and discharging) is to be implemented over the next time period Δt. At the next time instance, $\tau + \Delta t$, the EMPC is re-solved to again determine the optimal input trajectories.

2.2. DeepLynx and DeepLynx ORCA Adapter

For the RTO workflow, a DeepLynx ORCA adapter is being developed using the DeepLynx Python Software Development Kit [12]. The aim of this adapter is to automate the data communication through DeepLynx—without manual direction through the graphical user interface (GUI). DeepLynx constructs a container featuring a custom ontology, modifies that ontology, creates a data source, uploads the data, builds and runs type-mapping transformations, creates a time-series data source, and queries data. In this context, type mapping is a process that enables supplied data to be arranged into a graph structure within the container. A converted time-series table is attached—via a time-series data source—to a node in the graph. Detailed instructions on each type of operation are given in the wiki for DeepLynx's GitHub repository. Relationships among the data are represented as edges connecting the various nodes together. Each piece of time-series table data is stored in a node, and the nodes are connected via the edges representing the relationships between them. Though ontologies can be created from scratch, one called the Data Integration Aggregated Model and Ontology (DIAMOND) [13] already exists for NPPs and other relevant technologies in the nuclear application domain. DIAMOND, which includes various engineering-related metatypes and relationships, was implemented as the ontology for the RTO workflow.

The DeepLynx ORCA adapter supports the connection between ORCA and DeepLynx, as well as the connection between the physical system and DeepLynx. The adapter converts each entity's output files into the DeepLynx-preferred JavaScript Object Notation (JSON) format. With output files from each entity, as well as a graph plan file containing the node names and information on the relationship between the nodes, the adapter creates the structure for a graph-like database in DeepLynx. The output files of each entity are then automatically uploaded as a cumulative time-series table to a specific node in the database. The flow of data communication is shown in Fig. 2. In the first iteration of communication, a graph structure is created via type mapping, and a rule, governing how to attach a time-series table to a node, is determined. The remaining iteration serve only to upload or query data based on the information generated in the first iteration.

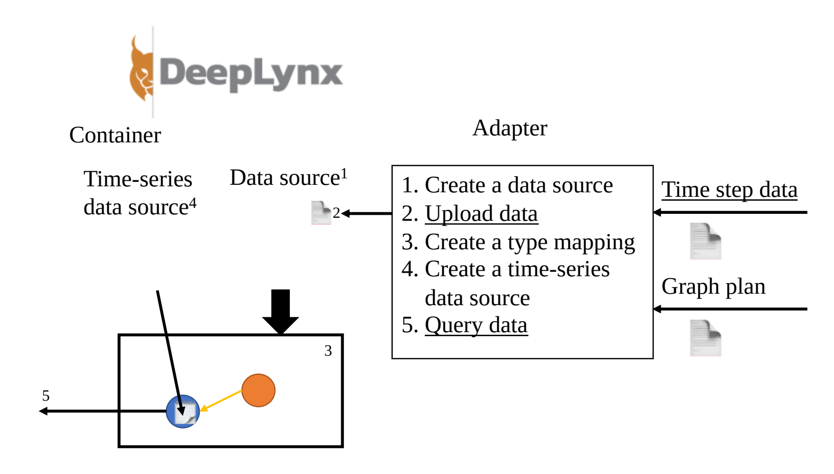

Figure 2. Data processing and storage workflow through DeepLynx.

3. VIRTUAL DEMONSTRATION SETTING AND RESULTS

3.1. Battery Model in Modelica

In this research, we modified the existing lithium-ion battery model [14], which had been developed in Modelica and can determine the SOC of a battery, based on the amount of energy charged/discharged. In the model shown in Fig. 3, random charging and discharging signals are given every 5 minutes to detect any changes in the battery SOC, which is determined by employing the following equation:

$$\text{Net charging rate} = \text{charging rate} - \text{net discharging rate}, \quad \textbf{(7)}$$

$$\text{Net discharging rate} = \min(\text{initial discharging rate, state of charge} \times (60/5) + \text{charging rate}), \quad \textbf{(8)}$$

where (60/5) is multiplied for changing unit from Wh to W. Fig. 4 shows changes in the normalized SOC of the battery, as detected via charging/discharging signals produced at random over a period of 600,000 seconds.

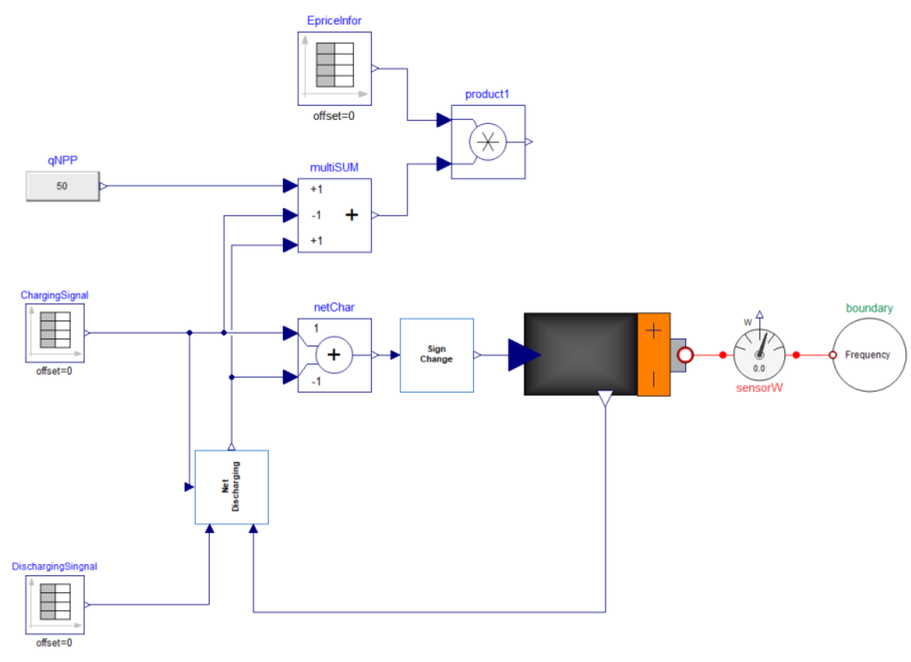

Figure 3. The Modelica model calculates the lithium-ion battery state of charge over time, based on charging/discharging signals. (Modification of the original model taken from [14].)

With the battery model in Modelica, we created a functional mock-up unit (FMU) and simulated, in FMPy, the information exchanges occurring among the battery model, DeepLynx, and ORCA. This work treats the FMU as a virtual representation of the Modelica battery model, which in the next study will be replaced by the actual physical system itself.

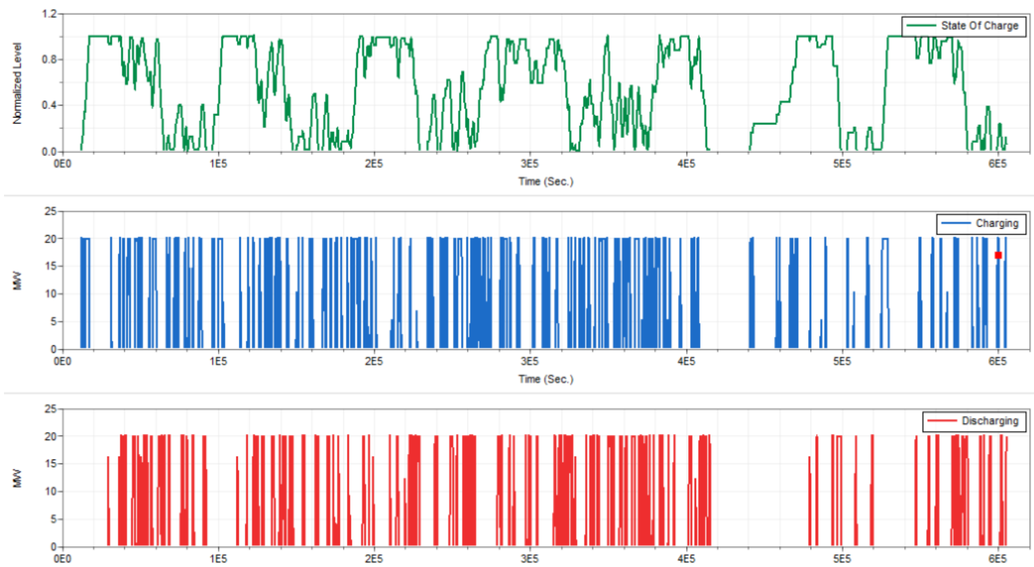

Figure 4. Lithium-ion battery state-of-charge changes corresponding to charging/discharging signals.

3.2. Real-Time Optimization Workflow Using ORCA

This section demonstrates the integrated RTO workflow with ORCA, along with the Dymola functional mock-up interface (FMI). (DeepLynx will be integrated later for the purposes of data transfer and information management.) A demonstration of an electrical storage device that utilizes the ORCA workflow was developed based on the parameters in Table I, using the naming convention given in Section 2.1. S_0, which was not defined in Section 2.1, represents the electrical storage device's initial SOC in MWh. Δt was selected as 5 minutes and charging/discharging actions are generated every 5 minutes. A sinusoidal function is used as a synthetic representation of real-time market locational marginal pricing (LMP). The prediction horizon t_{window} is chosen to be 100 minutes, where the calibrated grey-box model (Eq 1.) in ORCA show acceptable prediction accuracy.

Table I. Parameters used in the ORCA workflow.

Parameter		Value
Δt	Unit time horizon (minutes)	5
t_{window}	Prediction horizon (minutes)	100
γ_{RTE}	Round-trip efficiency during charge and discharge	0.8
q_{NPP}	Constant power generation of the NPP (MW)	50
S_{max}	Maximum energy capacity (MWh) of the battery	20
S_0	Initial state of charge of the electrical storage device (MWh)	0
q_{max}^C	Maximum charging power ratings (MW)	20
q_{max}^D	Maximum discharging power ratings (MW)	20

The dispatch optimization portion of the RTO workflow was set up as a linear programing (LP) problem, solved by the open-source optimization modeling language package Pyomo [15]. A sinusoidal price forecast model was used, with a LMP peak or bottom every 6 hours. Fig. 5 shows the optimal charging/discharging rates determined by ORCA. No charging or discharging was performed at the first LMP peak, since the electrical storage device begins in a depleted state. However, it charges when the LMP is relatively low at around the first bottom (~12 pm), and discharges near the second peak (~18 pm), when the LMP is higher. Another feature of this single-instance LP solve is that the electrical storage device completely discharges before the LMP reaches the bottom. Based on the anticipated LMP changes, the optimal solution is to leave no energy in the storage system, and to only start charging when nearing the LMP bottom. The SOC predicted by the ORCA grey-box model, as shown in Eq. 1, was also compared against the FMI outputs, with the grey-box model showing good agreement with the actual behaviors.

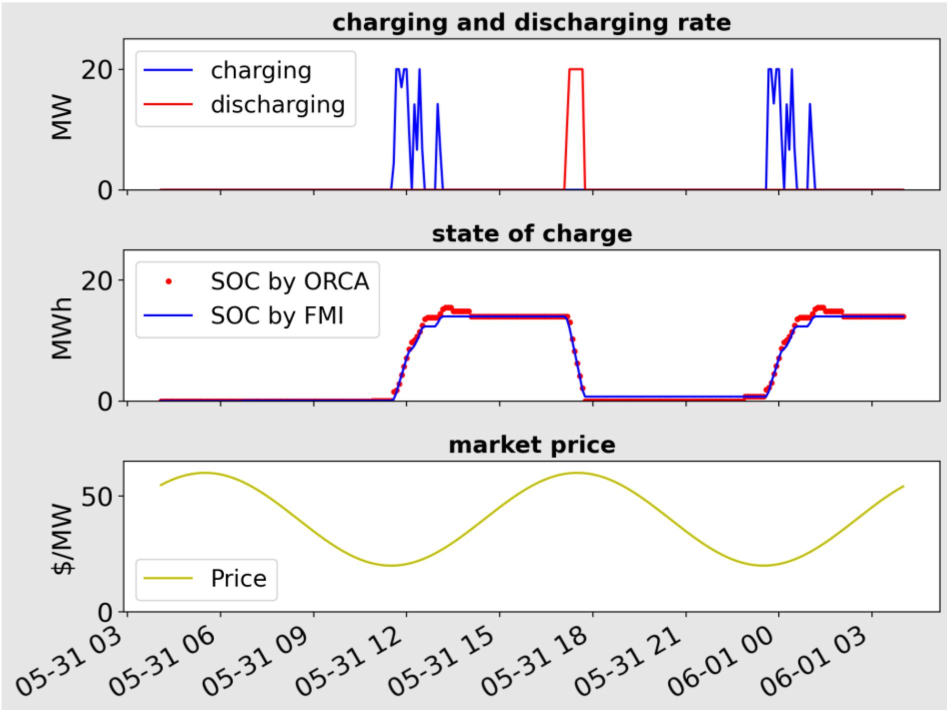

Figure 5. Transient of the optimal charging and discharging rate determined by ORCA. The predicted SOC is compared against the FMI outputs. Deviations are due to model errors in representing the actual battery model.

3.3. DeepLynx Integration

In this virtual demonstration, the FMU of the battery model in Modelica was developed as a virtual model of the physical system and ORCA communicating to each other via DeepLynx. Since a simple battery model was selected for the virtual demonstration, the inputs from the Dymola's Modelica model are the SOC and voltage, while the inputs from ORCA are the SOC and the charging/discharging power rates. The Dymola-side inputs are stored in a node called "Battery," whereas the ORCA-side inputs are stored in the other node, called "Control." These two nodes are connected using a "relates" relationship. The graph structure displayed on the DeepLynx GUI is shown in Fig. 6.

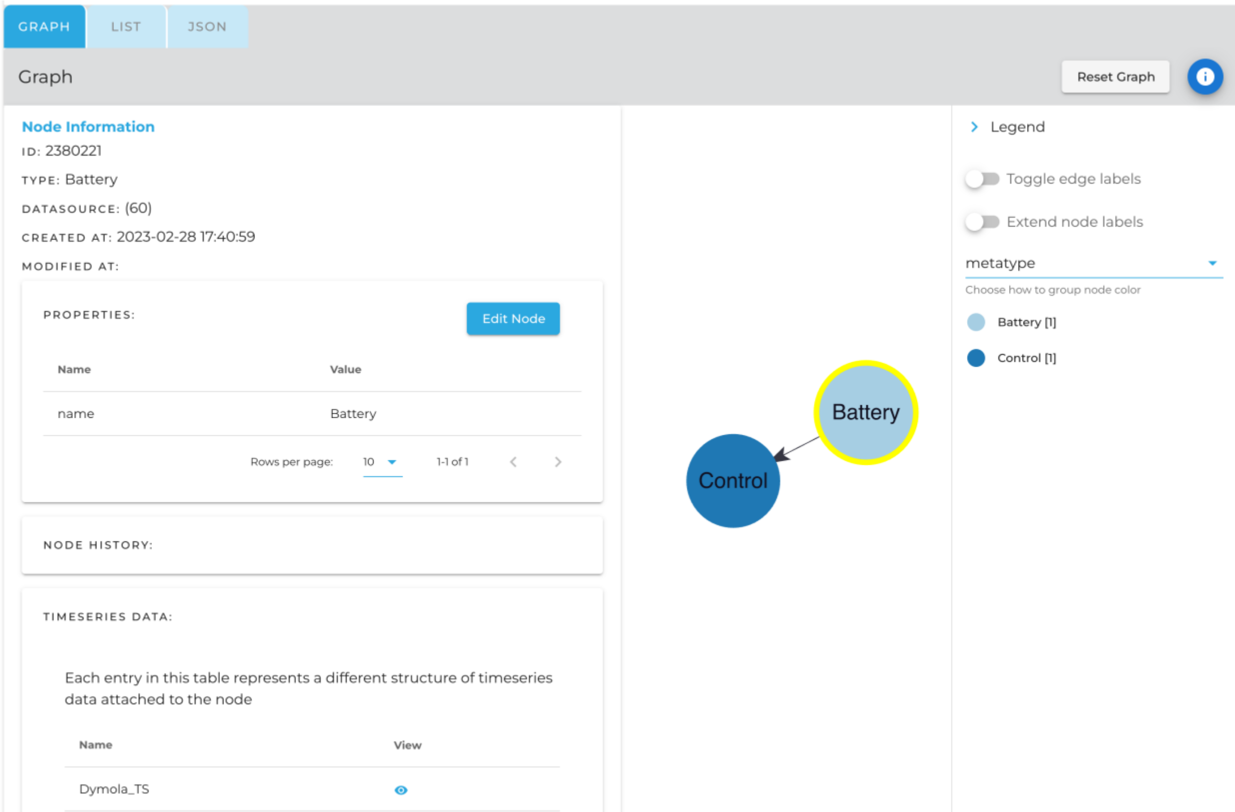

Figure 6. DeepLynx GUI data viewer, showing a battery node that includes a time-series table of Dymola output.

4. CONCLUSIONS

The RTO workflow and the behavior of key software (i.e., ORCA and DeepLynx) were confirmed via this virtual demonstration. Using the DeepLynx ORCA adapter, the virtual battery model sent virtual sensor data to DeepLynx, which then stored the data in graph structure in its Battery node. Queried sensor data were transferred to ORCA, which then returned optimal charging/discharging rates. These were, in turn, transferred to the DeepLynx and saved in the Control node. Data were then queried and loaded as the Dymola input. In this virtual demonstration, the physical assets took the form of a simple battery model, but the actual physical system is in fact more complicated. DeepLynx can create a more complex graph system [9], and the ORCA DeepLynx adapter can easily and systematically create a graph plan. Moreover, DeepLynx can intuitively modify the graph when displaying it on the GUI. In the future, the RTO workflow will be showcased using a physical IES. Meanwhile, ORCA workflow will be improved with nonlinear optimization algorithms. An appropriate IES will be selected and a DT created. The data connections must be set up, and the optimization structure should be established to enhance the right inputs while working with an external grey-box model of the system.

5. ACKNOWLEDGMENTS

This work was supported by the Integrated Energy Systems Program at Idaho National Laboratory under DOE operations contract no. DE-AC07-05ID14517.

6. REFERENCES

1. D. Krishnamoorthy, B. Foss, and S. Skogestad. "Steady-state real-time optimization using transient measurements." *Computers and Chemical Engineering*. **115**, pp.34-45 (2018). https://doi.org/10.1016/j.compchemeng.2018.03.021.

2. D. Mendoza, J. Graciano, F. Liporace, and G. Le Roux. "Assessing the reliability of different real-time optimization methodologies." *The Canadian Journal of Chemical Engineering*. **94** (3), pp.485-497 (2016). https://doi.org/10.1002/cjce.22402.

3. I. Miletic and T. Marlin. "On-line Statistical Results Analysis in Real-Time Operations Optimization." *Ind. Eng. Chem. Res*. **37** (9), pp.3670-3684 (1998). https://doi.org/10.1021/ie9707376.

4. P. D. Roberts. "An algorithm for steady-state system optimization and parameter estimation." *International Journal of Systems Science*. **10** (7), pp.719-734 (1979). https://doi.org/10.1080/00207727908941614.

5. A. Marchetti, B. Chachuat, and D. Bonvin, "Modifier-Adaptation Methodology for Real-Time Optimization." *Ind. Eng. Chem. Res.*. **48** (13), pp.6022-6033 (2009). https://pubs.acs.org/doi/full/10.1021/ie801352x.

6. M. Ellis, H. Durand, and P. Christofides. "A tutorial review of economic model predictive control methods." *Journal of Process Control*. **24** (8), pp.1156-1178 (2014). https://doi.org/10.1016/j.jprocont.2014.03.010.

7. V. Yadav, et al. "Technical Challenges and Gaps in Digital-Twin-Enabling Technologies for Nuclear Reactor Applications." TLR/RES-DE-REB-2021-17, U.S. Nuclear Regulatory Commission (2021). https://adamswebsearch2.nrc.gov/webSearch2/main.jsp?AccessionNumber=ML21361A261.

8. M. Tiller. *Introduction to Physical Modeling with Modelica*. Kluwer Academic Publishing, Boston (2001). https://books.google.com/books?hl=en&lr=&id=BbJRLKoTfwwC&oi=fnd&pg=PR9&ots=wMN_tMF3VK&sig=KZh73yTHHGL9D9bMB1xxlMzB80o#v=onepage&q&f=false

9. D. Garrett, T. Kajihara, J. Kim, and P. W. Talbot. "Real-Time Optimization Workflow Status Update." INL/RPT-22-69190-Rev000, Idaho National Laboratory (2022). https://doi.org/10.2172/1891509.

10. J. Darrington, J. Browning, and C. Ritter. "*Deep Lynx: Digital Engineering Integration Hub*." U.S. Department of Energy of Energy Office of Efficiency and Renewable Energy (EERE). Computer software (2020). https://doi.org/10.11578/dc.20200929.1.

11. C. Rabiti, et al. "*RAVEN User Manual*." INL/EXT-15-34123, Rev007, Idaho National Laboratory (2021). https://doi.org/10.2172/1784874.

12. J. M. Browning. "Deep-Lynx-Python-Package." U.S. Department of Energy Office of Science & Technical Information. Computer software (2021). https://doi.org/10.11578/dc.20210629.1.

13. A. Rashdan, J. Browning, and C. Ritter. "Data Integration Aggregated Model and Ontology for Nuclear Deployment (DIAMOND): Preliminary Model and Ontology." INL/EXT-19-55610, Idaho National Laboratory (2019). https://lwrs.inl.gov/Advanced%20IIC%20System%20Technologies/DIAMOND_Preliminary_Model_and_Ontology.pdf.

14. H. Wang, R. Ponciroli, and R. B. Vilim. "Development of Electro-chemical Battery Model for Plug-and-Play Eco-system Library." ANL/NSE-21/26, Argonne National Laboratory (2021). https://publications.anl.gov/anlpubs/2021/09/170551.pdf.

15. W. E. Hart, J. P. Watson, and D. L. Woodruff. "Pyomo: modeling and solving mathematical programs in Python." *Math. Prog. Comp*. **3**, pp.219-260 (2011). https://doi.org/10.1007/s12532-011-0026-8.