INL/RPT-22-69275
Revision 0

2022 FORCE Development Status Update

September 2022

Paul Talbot, Dylan McDowell, Joshua Cogliati, Botros Hanna, Gabriel J. Soto, and Han Bao

Idaho National Laboratory

IES
Integrated Energy Systems

INL/RPT-22-69275
Revision 0

2022 FORCE Development Status Update

Paul Talbot, Dylan McDowell, Joshua Cogliati, Botros Hanna, Gabriel J. Soto, and Han Bao

September 2022

Idaho National Laboratory
Integrated Energy Systems
Idaho Falls, Idaho 83415

http://www.ies.inl.gov

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

Page intentionally left blank

EXECUTIVE SUMMARY

Integrated energy systems (IES) have emerged as a promising approach to increasing the expected economic efficacy of both existing and advanced nuclear power plants (NPPs) in the face of increasing load variability due to non-dispatchable variable renewable energy (VRE) sources such as solar and wind. However, the economic benefit of IES integrations is dependent on market characteristics, electricity demand, and weather patterns. Traditional grid portfolio analysis software was not designed to capture critical attributes such as load switching, variability, secondary commodities, and the high degree of uncertainty in hourly demand profiles. This gap in capability led to the development of the Framework for Optimization of Resources and Economics (FORCE) tool suite, including the long-term grid portfolio planning tool Holistic Energy Resource Optimization Network (HERON) and the transient process model analysis library HYBRID, as well as other supporting libraries.

Many high-level, low-resolution calculations can be quickly performed using simple spreadsheet tools or expert engineering judgement. These fundamental calculations can inform evaluations of the value of performing more costly high-resolution calculations. These low-resolution calculations do not capture some elements that are potentially critical to understanding economic opportunities, such as high levels of demand uncertainty/variability. Ignoring these elements risks inaccurately representing the behavior of time-integrated components such as energy storage and IES mode switching, including switching from power generation to integrated secondary commodity (e.g., hydrogen or desalinated water) production.

It is for this purpose that the IES program continues to invest in free open-source tools for IES analysis via the FORCE tool suite. FORCE tool development is driven by three software characteristics: capability (i.e., the ability to perform specific analysis), accessibility (i.e., the barrier of entry for analysts to use the tool suite), and robustness (i.e., the consistency and reliability of the tool suite in solving problems within its scope).

In fiscal year (FY)-2022, significant effort was invested to improve the capabilities of HERON and HYBRID so as to address questions posed by analysts within the IES program, partner programs at national labs and universities, and other external efforts with industry partners. Effort was also made to improve accessibility, enhance cross-tool communication, and expose more optional features to analysts. Finally, robustness was improved by updating FORCE tools to be compatible with their dependent libraries, and improving parallel computation performance on high-performance computing (HPC) clusters.

While FORCE tools are beginning to reach a quality level that enables broader use by energy analysts, the tool suite is the product of research software engineering, which by its very nature aims to deliver solutions to specific problems while attempting to remain generic and modular enough to solve similar problems in the future. Inasmuch as the FORCE tool suite is envisioned as a holistic approach to solving future energy problems, effort must continue to be made to move the tools from targeted application demonstrations to reliable software that does not require developer interaction for effective use.

Page intentionally left blank

CONTENTS

FIGURES

TABLES

Page intentionally left blank

ACRONYMS

ARMA	autoregressive moving average
CSV	comma-separated values
DISPATCHES	Design Integration and Synthesis Platform to Advance Tightly Coupled Hybrid Energy Systems
DOE	U.S. Department of Energy
DOF	degree of freedom
FARM	Feasible Actuator Range Modifier
FMI	functional mock-up interfaces
FMU	functional mock-up units
FORCE	Framework for Optimization of Resources and Economics
FPOG	Flexible Plant Operation and Generation
FY	fiscal year
GUI	graphical user interface
HERD	HERON Runs DISPATCHES
HERON	Holistic Energy Resource Optimization Network
HPC	high-performance computing
IDEAS	Institute for the Design of Advanced Energy Systems
IES	integrated energy systems
INL	Idaho National Laboratory
IRR	internal rate of return
JSON	JavaScript Object Notation
LMP	locational marginal price
LWRS	Light Water Reactor Sustainability
MOPED	Monolithic Optimizer for Probabilistic Economic Dispatch
NPP	nuclear power plant
NPV	net present value
PEM	proton-exchange membrane
PI	profitability index
RAVEN	Risk Analysis Virtual Environment
ROM	reduced-order model
RTE	round-trip efficiency
TEAL	Tool for Economic Analysis
VRE	variable renewable energy
XML	Extensible Markup Language

Page intentionally left blank

2022 FORCE Development Status Update

1. BACKGROUND

Due to the rapidly changing nature of the energy landscape in both the U.S. and abroad, including the increase in non-dispatchable variable renewable energy (VRE) deployment and the abundance of carbon-based dispatchable fuels, the Integrated Energy Systems (IES) program under the Department of Energy (DOE) has been engaged in determining whether IES could assist in providing nuclear power plants (NPPs) with a viable strategy for achieving future economic competitiveness. This includes enhanced dispatch flexibility for NPPs, thanks to thermal and electrical storage technology as well as the generation of ancillary commodities such as hydrogen, desalinated water, and synthetic fuels from the heat generated by NPPs. The technical and economic viability of these systems is of primary concern to the IES program, with the intent of providing insight to the wider nuclear energy community at large.

To comprehend the potential viability of these complex integrated systems, new approaches to modeling and analysis have been required. Three main gaps that existed in energy analysis ultimately led to the design and formulation of analysis tools for IES. These gaps were the traditional approaches to dispatchable energy analysis (approaches that did not capture the variable and non-dispatchable VRE technologies), traditional analysis' focus on single-commodity electricity markets, and low-uncertainty analysis of traditional dispatchable systems. In addition, existing tools for technical analyses of NPP performance did not capture the intricacies introduced by tightly coupling the NPP to other processes, requiring additional analysis tools to understand the technical and safety limitations of such couplings.

1.1 Analysis Workflows

The primary task in IES modeling and simulation is to predictively determine the potential technical and economic possibilities for new IES configurations in target areas, with a particular focus on advanced nuclear power generation. Many questions must be answered when considering technical and economic viability. What economic stresses do NPPs experience in a given target region? Which emerging technologies are changing the energy landscape? What types of technologies might serve as viable IES to improve the economic outlook for NPPs? Each of these questions can be answered (via various levels of resolution) to gain a complete picture of the opportunities for nuclear energy in a given region.

For example, consider a case in which an IES that produces a secondary commodity (e.g., hydrogen) is of interest to a NPP owner/operator. Some initial questions must be considered. What is the expected annual average market size and price for hydrogen in that region? (While the details of the market are not needed, order-of-magnitude estimates should be considered before engaging in detailed analysis.) What is the cost of introducing this technology to a new or existing NPP in that region? Once again, order-of-magnitude estimates allow for quick evaluation of whether there is significant reason to perform a more detailed analysis. If the estimated potential profits outweigh the estimated technology introduction costs, the question can then be pursued at greater resolution. This kind of estimate is relatively easy to perform, and can be accomplished by collecting a few values in a spreadsheet application.

If a new technology meant for deployment is a type of energy storage system, such estimates become increasingly difficult. What is the potential value of arbitrage (e.g., the cumulative potential of storing electricity during periods of cheap electricity prices and then releasing the stored electricity during high-price periods)? Some initial estimates may be possible based on existing historical data to confirm whether introducing energy storage could increase economic viability despite the cost of installing the energy storage technology. These estimates may require some simple scripting or formula, but remain straightforward from a modeling standpoint.

Once a reasonable expectation of economic viability is determined, careful modeling of the steady-state and transient operation of the IES components is necessary to establish the physical operating limitations of the technology. Process models determine the IES system's limiting characteristics, which can then be applied to a more detailed long-term optimization planning process. This analysis requires robust and intricate modeling to correctly capture technological limitations and transient behaviors, and is well beyond the scripting required for the initial estimates. Model toolsets such as HYBRID (see below) are well-suited to this type of analysis.

Similarly, the markets of interest must be characterized. How are the various commodity markets, including electricity and hydrogen, projected to evolve over the next several decades? What drivers determine this evolution? These data are critical for model accuracy when conducting a long-term planning analysis. Various forms of market modeling tools can help answer these questions, but frequently require software that goes beyond simple scripting and relies on expert knowledge for effective use. IES partners at several national laboratories are experts in this area, providing invaluable contributions to the characterization of potential markets.

Uncertainty regarding the availability of non-dispatchable energy sources such as solar and wind, along with the uncertainty in the demand, should also be captured. While portfolio optimization can be performed on a single dispatch, such analysis provides no guarantee that the chosen portfolio will be optimal in general, let alone specifically in regard to risk aversion. With significant economic consequences to under- or overproducing energy, understanding the possible scenarios that a portfolio might encounter is necessary for understanding the statistical merit of that portfolio. Uncertainty analysis toolsets such as the Risk Analysis Virtual Environment (RAVEN) are ideal for exposing potential scenarios based on historical training data, and for creating synthetic histories that can challenge energy portfolios so that their economic merit can be accurately represented.

Finally, with the technical, market, and stochastic elements characterized, long-term economic optimization can be performed. Long-term planning that correctly models energy storage, secondary commodities, and uncertain scenarios generates robust results regarding optimal portfolios, and lays the groundwork for follow-up-specific engineering analysis. This kind of analysis is complex, tying together several other types of complex analyses. However, this complexity is worth solving due to the high-impact nature of the solutions obtained. Determining the optimal configurations of energy portfolios—having considered the system's uncertainty, markets, and technical limitations—and further capturing the inertial behavior of storage systems and ramping units provides much more robust answers to questions pertaining to economic potential and viability than can be obtained via the loose scoping estimates conducted prior to the complex analysis. The complex analysis leverages insight into the time-dependent intersection of multiple commodities in multiple markets, encompassing a level of decision making that dwarfs traditional power system modeling in terms of degrees of freedom (DOFs). Tools such as the Holistic Energy Resource Optimization Network (HERON) (see Figure 1) are designed to make this kind of complex analysis approachable for energy analysts, requiring minimal effort.

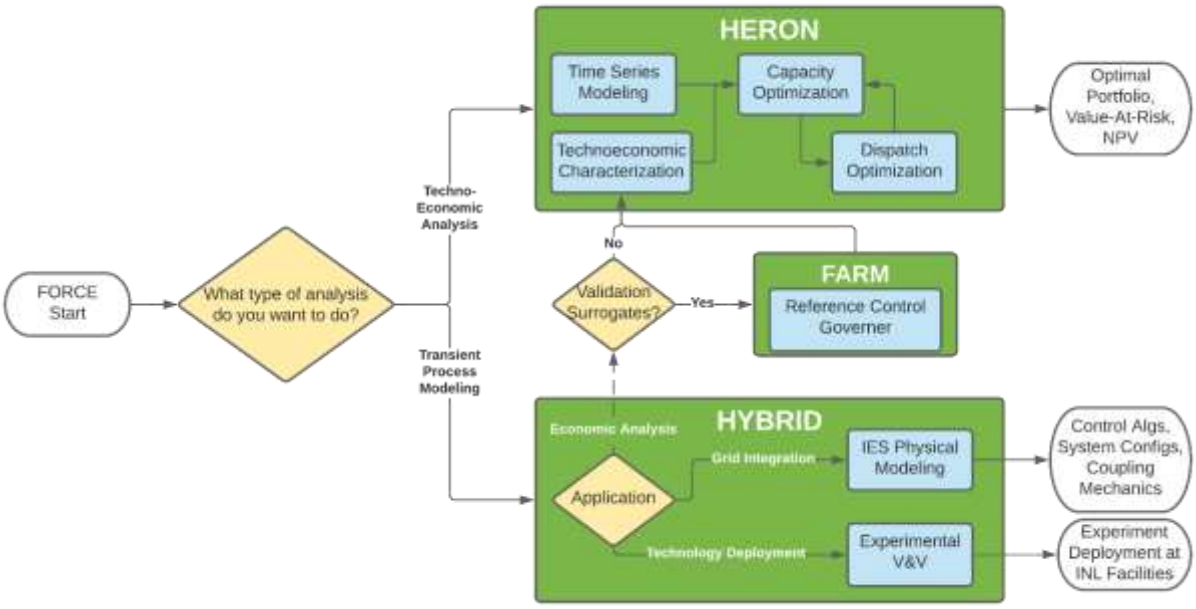

Figure 1. FORCE workflows from HERON and HYBRID provide a variety of calculations surrounding IES configurations, from transient process modeling to cost analysis and long-term economic planning.

1.2 The FORCE Framework

The Framework for Optimization of Resources and Economics (FORCE) (Figure 2) is the tool suite developed and maintained by the IES program in order to aid in investigating the technical and economic viability of new IES configurations, especially those involving NPPs. In collaboration with the DOE Light Water Reactor Sustainability (LWRS) program's Flexible Plant Operation and Generation (FPOG) Pathway, the IES program develops these tools to support analysis at U.S. national laboratories, as well as to support industry and university studies. While LWRS FPOG focuses on existing nuclear reactors, the IES program focuses on advanced reactor technologies.

Figure 2. FORCE is a software toolset developed and maintained by the IES program.

Within the FORCE tool suite are two flagship simulation tools: HERON and HYBRID (see Figure 3). The HERON software targets stochastic optimization of energy-grid portfolios, on the decades time scale, in evolving multi-commodity markets. HERON typically performs hourly dispatch optimization to guide portfolio design optimization in a stochastic sense, capturing the time-continuous effects of storage and technology ramping limitations. HYBRID is a library of technical and economic process models, with associated workflows for validation, verification, surrogate model generation, and deployment. HYBRID typically performs analyses on a second-by-second time scale, focusing on the transient dynamics of the coupled technologies.

Figure 3. HERON and HYBRID are the two flagship software tools in FORCE.

Analyses conducted using HYBRID or HERON are naturally intended to inform analyses across FORCE. For example, steady-state and transient analysis in HYBRID provides values for conversion efficiencies and production ramping limitations—values usable in HERON. Similarly, dispatch set points from HERON analyses can be used to investigate IES performance.

1.3 FORCE Support Software

In addition to the main workflows of HERON and HYBRID, a variety of additional software libraries support FORCE analysis as well. Most FORCE users need only be critically aware of HERON and HYBRID; other software in FORCE are treated as under-the-hood libraries that users should not need to interact with frequently.

RAVEN is a software package that was designed at Idaho National Laboratory (INL) for risk and uncertainty analyses. It allows complex workflows to be constructed and then maintained to perform specific uncertainty-related analyses. In the context of HERON, this includes training synthetic history generators for signals such as energy demand, wind speed, and solar availability. Synthetic histories of these signals allow HERON to sample many possible scenarios for dispatch optimization to consider, thus affording an aggregate comprehension of how well a configuration might behave statistically, rather than deterministically for a single forecast. Furthermore, RAVEN provides uncertainty-informed optimization strategies for optimizing capacity, given the statistical dispatch behavior. In HYBRID, RAVEN is used to construct validation workflows to ensure alignment between various process models and the available experimental or literature data. RAVEN also provides mechanisms to train surrogate models, reduced-order models (ROMs), machine learning and artificial intelligence models, and similar digital twins that accurately represent trained models but at a vastly reduced computational overhead. These digital twins can then be serialized as advanced models that can be inserted directly into optimization routines such as those used in HERON.

The Feasible Actuator Range Modifier (FARM) software bridges the gap between the complex transient process models in HYBRID and the streamlined models in HERON. For the sake of computational tractability, HERON uses component models that are reduced to input-output relationships. For example, HERON may represent a hydrogen production component as consuming steam and electricity to produce hydrogen, all in ratio. However, there are many additional factors to consider in the component's operation, such as internal temperatures and pressures, of which HERON has no knowledge. FARM provides in-loop validation tools whereby the manifest variables (e.g., steam, electricity, and hydrogen) can be projected to the latent variables (e.g., temperature and pressure). This validation can impose additional operational constraints on dispatch optimization and provide a feedback mechanism much more quickly than could a full physics model.

The Tool for Economic Analysis (TEAL) software is a simple library package for calculating various economic metrics, based on related activity over a given period. While TEAL can be used as a standalone tool in RAVEN, it is used directly in HERON as an economic metric calculator. Economic metrics of interest from TEAL include net present value (NPV), profitability index (PI), internal rate of return (IRR), and levelized cost.

1.4 State of the Tool Suite

Prior to the start of fiscal year (FY)-2022, the existing tools that are now under FORCE were largely independent. While TEAL and RAVEN had been somewhat stable previously, HERON experienced its first major update late in FY-2021 and had been almost entirely disconnected from HYBRID. FARM had been used to demonstrate some very simple workflows, but not integrated into any system analysis activities. FORCE itself had been identified as a term to describe a collection of tools, but with no structure or supporting software.

Priorities for evolving the FORCE tool suite fall into three major priority areas:

- Capability – Above all else, the FORCE tool suite is designed to accurately solve analysis questions pertaining to IES. As "all models are wrong but some are useful," we seek to connect useful models that provide analysts with accurate decision-making metrics. Increasing capability is driven by the needs of analysis within IES or our partners.
- Accessibility – For the FORCE tool suite to be useful to analysts, it should be accessible. Time spent learning to use the tools in the suite and struggling with its structure is time not spent on analysis; thus, there is a balance between the time it takes to develop accessibility features (e.g., user interfaces, input editors, plot generators, and install packages) and the time analysts spend struggling without those accessibility features. That balance becomes more pronounced when external collaborators seek to use the tool suite without in-house expert guidance.
- Reliability – In research software engineering, it is common to develop a tool or process to solve a particular problem, then not extend it to solve similar problems. For example, a tool might be developed to solve IES dispatch in a regulated market without storage for one case, and in deregulated markets with storage for another, but be unable to solve dispatch in a regulated market with storage. Because development is driven by analysis and application rather than software requirements, there are often gaps in a tool's performance. Reliability features ensure that a research tool is available, performs consistently, and can solve the problems within its scope.

With these three priority areas in mind, new development activities were selected based on requirements for the current year's analysis cases, reported needs from internal users, and reported needs from external users. The following prioritized development activities are discussed in the following sections:

- Functional mock-up interface (FMI) and functional mock-up unit (FMU) development and roadmapping
- TEAL visualization improvements
- Parallel communication and operations maintenance
- FORCE tool intercommunication
- DISPATCHES integration
- HERON improvements.

Other activities that were considered but deemed beyond our bandwidth to implement over the last FY include:

- Comprehensive installation package for FORCE tools
- Graphical user interfaces (GUIs) for:
    1. FORCE, to guide analyses
    2. FORCE, for installation
    3. HERON, for long-term economic planning
    4. HYBRID, for validation workflows
    5. RAVEN, for training synthetic history generators
- Transition from the autoregressive moving average (ARMA) ROM in RAVEN to the more robust SyntheticHistory ROM
- Multi-resolution dispatch optimization in HERON (seasonal, day-ahead, real-time)
- Capacity expansion modeling in HERON.

These may still be considered in future work.

1.5 FMI and FMU Developments and Roadmapping

A robustly working FMU would allow improved model sharing, both from RAVEN to other programs and vice versa. This goal led to testing and demonstration of the FMU capability in FORCE. Development work and roadmapping was done on FMUs and FMIs in FY22. Prior to the start of FY22 FMU capabilities had only been demonstrated at a very proof-of-principle stage and were not integrated into FORCE. Now a limited FMU creation capability has been integrated into RAVEN. This development better illustrates which features can be expected to be used and which will be more difficult to integrate with FORCE.

1.6 TEAL Visualization Improvements

TEAL is a FORCE tool whose primary purpose is to calculate economic metrics. TEAL can be run as a standalone Python tool, a model plugin for RAVEN, or as a Python library. The latter approach allows other software to build cashflow objects and calculate common economic metrics such as NPV, PI, and IRR. It also allows for calculating cost targets such as levelized cost. TEAL has been further extended by work in other programs this year in order to provide Pyomo-ready expressions for economic metric calculations, enabling algebraic expressions to be constructed and optimized.

TEAL is used within HERON to evaluate cashflows that are the economic drivers in both dispatch and capacity optimization. While NPV is the only metric currently used extensively in analyses, the other metrics may be of future interest.

Section 2.3 documents the efforts in developing the TEAL output visualization capability. In the past, outputs (e.g., various cashflows) were generated and printed to screen. Now users can either consider these outputs in comma-separated values (CSV) format or visualize them in various ways. Yearly cashflows can be plotted in either stacked or grouped bar charts, covering different time ranges. Stacked charts can be used to compare the sums of inflows and outflows, and the grouped charts can easily show the differences between individual flows. Net cashflows can also be visualized in two ways: yearly and cumulative. The yearly net cashflow indicates the annual changes to the net cashflow in each project year; the cumulative net cashflow represents the annual changes to the cumulative net cashflow over a selected time range. In addition, the cumulative inflows/outflows can be visualized and compared in a donut chart. Their component names and values determine their color assignments and donut portions.

1.7 Maintenance

Maintenance is needed as part of the use, adaption, and development of software, due to changes to both the requirements of the users and the environment the software runs in. FORCE runs in a high-performance computing (HPC) environment and uses multiple libraries, both of which subject FORCE to changing software and hardware. Thus, there are expected changes to which FORCE must adapt. FORCE tools support desktop and HPC analysis. For running on HPC, they support various ways of running in parallel, including shared memory and distributed memory. Running on HPC required efforts to fix and improve the parallel running.

1.8 FORCE Tool Communication

Maturing the FORCE suite requires interconnections between the various FORCE tools in order to ensure consistent analysis of the technical and economic potential of different possible IES configurations. An interface that facilitates access to several tools in the FORCE suite is necessary, since each tool individually answers a portion of the problem (as demonstrated in Section 1.2 and Section 1.3). In prior work, the analyst had to transfer data/models between the FORCE tools *manually*. Manual data transfer leads to errors and confusion about units of measure. In addition, it is a time-wasting process, especially if the analyst must transfer the data frequently when performing multiple simulations for sensitivity analysis. The process of manual data transfer includes data processing, data transfer between different file types, checking the consistency of the units of measure, excluding irrelevant data, renaming the variables, and making assumptions about incomplete data.

While the goal is to completely integrate all the FORCE tools introduced in Section 1.2 and Section 1.3, we prioritized vertical integration of HYBRID and HERON (see Figure 4), since HYBRID and HERON were already used together in previous studies (see [1], [2], and [3]), and there is a need to automate data exchange between them.

Figure 4. HYBRID-HERON vertical integration and data exchange.

HYBRID and HERON work side by side (see Figure 4) in the following manner. HERON analyzes the long-term viability of potential IES configurations, while HYBRID's models provide a high-resolution short-term analysis. HYBRID models determine the fundamental properties of the IES components. These fundamental properties include transfer functions, operational ramping limits, and economics. These properties are used in HERON as constraints and economic drivers for long-term energy portfolio optimization. On the other side, the feasibility of the optimized energy dispatch, as calculated by HERON, is analyzed using HYBRID to ensure no physical or operational constraints are violated. Recent developments in automating the data exchange between HYBRID and HERON are presented in Section 2.5.

1.9 DISPATCHES Integration

The Design Integration and Synthesis Platform to Advance Tightly Coupled Hybrid Energy Systems (DISPATCHES) is an open-source Python software package developed to optimize IES for operation within a power grid system via energy market signals [4]. DISPATCHES was developed primarily by the National Energy Technology Laboratory, with contributions from INL, Lawrence Berkeley National Laboratory, the National Renewable Energy Laboratory, and Sandia National Laboratory.

DISPATCHES and HERON both solve the same problem—stochastic optimization of capacity and energy dispatch, based on market signals—but do so in different ways. HERON generates a RAVEN workflow that solves the problem using an "outer-inner" approach: it solves an outer optimization over capacities and, at every step within the capacity parameter space, conducts an inner optimization for energy and product hourly dispatch over a user-specified number of stochastic histories or scenarios. By contrast, DISPATCHES solves the problem via an "all-at-once" monolithic solve, meaning that capacity variables are optimized at the same time as the dispatch variables for all sampled stochastic histories. The DISPATCHES approach works well for problems of limited size and project length, since the added number of variables and constraints for the singular optimization level require increased computational resources.

The HERON and RAVEN duo create and solve generic models for user-defined components that can then be integrated with more robust transient models in Modelica by using HYBRID (within the FORCE tool suite) or linking to other external models. Dispatch optimization in HERON and RAVEN is based on generating algebraic expressions using Pyomo, with constraints and objective functions being programmatically generated based on user input. DISPATCHES, on the other hand, relies on steady-state solutions of algebraic physics models from the Institute for the Design of Advanced Energy Systems (IDAES) platform. Currently, DISPATCHES houses three sample IES cases: a fossil fuel case, a renewables case, and a nuclear case. The IES nuclear case solved by the DISPATCHES software entails a fixed-output NPP with an electricity split-flow to the grid for direct sales as well as to a proton-exchange membrane (PEM) electrolyzer for conversion into hydrogen [5]. The hydrogen is stored in a hydrogen tank from which it can be sold to a hydrogen market for profit. Alternatively, the hydrogen from the tank can be sent through a hydrogen turbine to be converted back into electricity at opportune times via combustion [6]. A flow diagram of this process is shown in Figure 5. Within the IDAES platform are Pyomo algebraic models for the NPP, PEM electrolyzer, hydrogen tank, and hydrogen turbine; DISPATCHES connects all resource streams through Pyomo connectors, arcs, splits, and property tables.

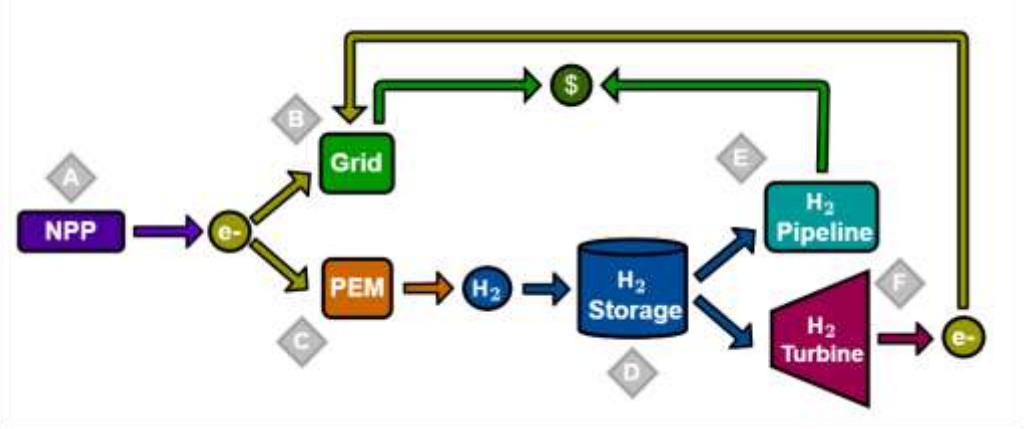

Figure 5. Diagram of DISPATCHES nuclear case components and the exchange of resources within the model.

The core of the DISPATCHES workflow relies on flowsheets, which are Python methods that define the physics of the problem through IDAES models, including all required variables and constraints. Flowsheets represent the model at a specific point in time. Accompanying the flowsheets are an initialization method and a method to "unfix" the DOFs of the problem. The initialization method initializes all variables, initially fixing some necessary parameters (e.g., NPP power output). The "unfix" method is meant to be more customizable, allowing users to augment the DOFs of the problem (e.g., the split fraction on the NPP electricity outlet). It also allows users to introduce upper and lower bounds on variables. The customization is hard coded, in that the user must implement changes via an overloaded method. HERON carries the potential to provide automatic customization based on component input from users.

The actual DISPATCHES workflow is conducted via Jupyter notebooks. Currently, two main workflows are available within the DISPATCHES repository: a multiperiod and double-loop optimization for each of the three IES cases. The double-loop optimization should not be confused with the HERON/RAVEN outer-inner optimization—here, "two loops" refers to a day-ahead market and a real-time market looping optimization. Current integration efforts are focused on multiperiod optimization of the nuclear IES case. Multiperiod optimization represents the all-at-once optimization strategy for the IES case. The optimization variables are the component capacities (for the PEM electrolyzer, hydrogen tank, and hydrogen turbine) as well as the resource dispatch to the electricity and hydrogen markets (and between components, as necessary) per unit time. The dispatch is driven by market signals in the form of local marginal prices (LMPs) provided within the DISPATCHES repository. The LMPs are given hourly in 24-hour series. In the given signal data, there are two clusters of 24-hour time series per year, and a total of 10 years of data per scenario. A collection of five sampled scenarios can be used within the nominal DISPATCHES workflow. The multiperiod notebook iteratively creates instances of the flowsheet for every 24-hour time series per cluster, cluster per year, year per scenario, and scenario per user specification.

The FORCE tool suite benefits from greater flexibility by adding a new optimization method that can also utilize the IDAES algebraic models and the monolithic optimization solving strategy, both provided by DISPATCHES. The DISPATCHES optimization approach, though more memory-expensive than the standard HERON optimization through RAVEN, works better when applied to problems of limited scope and short project length (dependent on computational limits). Integrating HERON and DISPATCHES will give users the choice of optimizing via the outer-inner or the all-at-once approach. This gives users the flexibility to solve design problems by using the best-suited optimization method. It also allows them to automatically incorporate certain RAVEN features into the DISPATCHES workflow (e.g., automatic generation of cashflows and economic metrics through the TEAL plugin, as well as ROM creation and sampling through RAVEN). There may be potential interest in using HYBRID to link dispatch optimization solutions from the steady-state DISPATCHES solutions to some transient models.

The goal is automatic generation, through HERON, of DISPATCHES workflows as an alternative to generating RAVEN workflows. For this milestone, we aimed to give a simple demonstration of a DISPATCHES workflow generated via HERON. Using a HERON XML input, we can closely model the components provided by the nuclear IES case within DISPATCHES. Upon running HERON, we can provide component information to the DISPATCHES flowsheets and build a multiperiod model that solves for the optimal capacities of the PEM electrolyzer, hydrogen tank, and hydrogen turbine, based on multiple market pricing scenarios and the dispatching according to those scenarios. Additionally, we wanted to show integration of TEAL cashflows in the objective function of the optimization.

1.10 HERON Improvements

The HERON software comprises the economic and stochastic analysis portions of the FORCE tool suite. HERON provides analyses on the long-term economic feasibility of IES, while considering systematic uncertainty in market demand. During FY-22, several new features and improvements were added to HERON to enhance user accessibility and capabilities. This section details the premeditated work we identified in planning for FY-22, along with the motivations behind these changes to HERON.

In recent years, the HERON user base has grown profoundly. Analysts and researchers have adopted HERON and FORCE as their primary tools of choice when analyzing potential energy systems. With this growth, we have received feedback from users and have identified several features that could improve the workflow for future analyses. A common piece of feedback reported was the desire to decrease the time it takes to verify the behavior of a HERON simulation.

Due to the design and computational complexity of some simulations, it can take valuable time to verify and assess the performance of a HERON workflow. The development team has sought to reduce this time by implementing features that afford the user greater insight into the behavior of their simulation during the prototyping phase of analysis. One major change identified for FY-22 was the capability of the user to specify static CSV files in place of using serialized ARMA models for synthetic history generation. This feature will allow users to skip the initial step of training an ARMA model using RAVEN, and instead directly employ data from a particular source.

Figure 6 visualizes HERON's mode of computation as a pair of coupled loops. Traditionally when users employ an ARMA model as the primary source of stochastic input data, the inner loop is run multiple times using a perturbed synthetic time series. This provides the optimizer with a varied set of histories, and considers the uncertainty inherent in energy-demand data. With the added capability of using static CSV files, the mode of computation is flattened to a single loop. Thus, the user sacrifices the statistical robustness of using many ARMA samples in favor of the speed of using a single deterministic time series. This way, one can quickly verify the simulation using a CSV, then, when ready to run a full simulation, it could be replaced with an ARMA model.

Another method of computation was also added to HERON's capabilities during FY-22. Similar to the monolithic approach of DISPATCHES, the new Monolithic Optimizer for Probabilistic Economic Dispatch (MOPED) workflow can solve a HERON workflow through single-level optimization. This significantly reduces the simulation length for non-complex systems. Section 2.7.2 details how this new workflow is used, and points out its limitations.

Another required addition to HERON during FY-22 was the identification of allowing users to set specific RAVEN settings within a HERON XML input file. It was determined that if a user could specify parallelization and optimizer settings instead of relying on the defaults traditionally provided by HERON, accessibility would be increased. While small, these changes are expected to profoundly impact ease of use and future analyses. In addition to RAVEN-specific settings, the value of adding the capability to optimize variables not necessarily associated with component resource capacities was recognized. For example, an analysis would typically seek to optimize the size of a NPP; but if we wanted to optimize the market bid adjustment of the NPP, addition of this feature would allow for that scenario.

Other work that was planned for FY-22 but remains uncompleted due to a lack of available developers was the migration from ARMA-generated synthetic histories to RAVEN's more modular SyntheticHistory system. This new system allows users to employ many time-series analysis algorithms instead of solely relying on an ARMA model for synthetic history generation. A lot of work was done to complete this migration, but as of this writing, it is not yet 100% complete. The final few changes required to complete this task are relatively minor and should be completed before the end of FY-22.

Finally, training a synthetic history can be described as both an art and a science. Analysts use discretion when determining if their history is overfitted or overgeneralized. This leads to an iterative development cycle in which an analyst tries a specific set of input parameters and continually tweaks them until arriving at a satisfactory history. As was mentioned, the developers' goal is to reduce the time it takes to iterate through a HERON workflow. It was identified that having an interactive way to train synthetic histories by using a Jupyter notebook would be helpful to our users. This feature was not completed during FY-22 but is planned as a follow-on activity during FY-23. Currently, there are development plans to eventually implement a GUI for FORCE. This would eliminate the need for a Jupyter notebook, but will improve users' lives in the meantime.

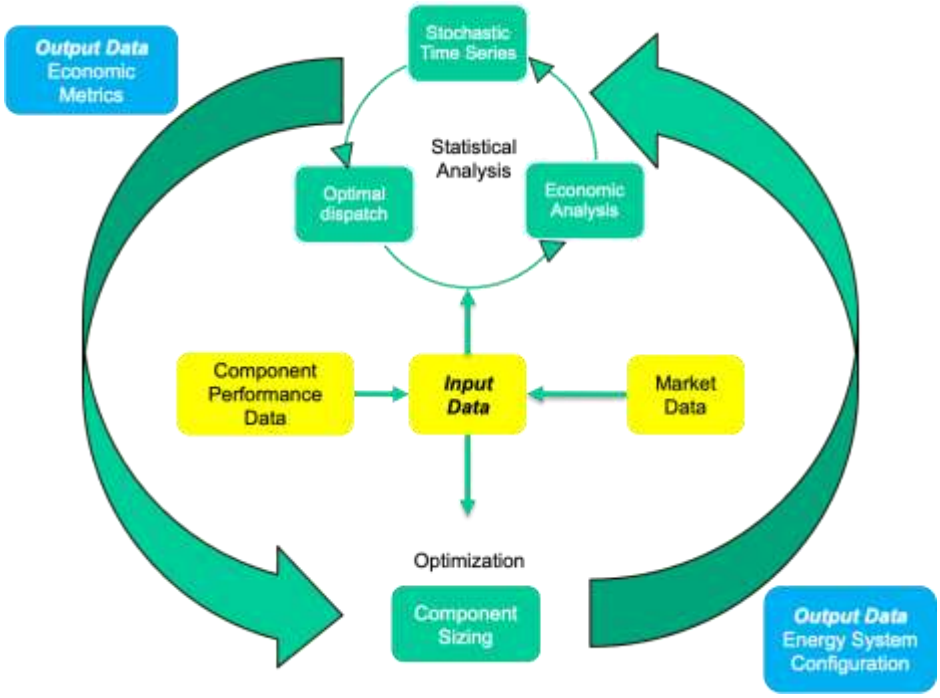

Figure 6. HERON's computational pathway.

2. DEVELOPMENT AND DEMONSTRATION

2.1 Introduction

In this section, we consider the different areas of improvement in the software as a result of this year's activities. Note that the developments we pursued were either identified as program objectives or as discovered gaps from the analysis performed by the IES program, such as in the three so-called use cases: carbon conversion, synthetic fuels, and thermal energy storage. Each of these use cases uses or will use both pathways in the FORCE suite.

As such, the most significant demonstrations resulting from the development work this year was the successful analyses performed or planned as part of these use cases. In this document, we focus on the improvements made, and give simple demonstrations of the benefits thereof. We encourage interested readers to observe these improvements in action in the demonstration cases released this year by the IES program, or that will be released in coming years.

As described above, FORCE tool development is accomplished via an outcome-centric process. Capability, accessibility, and robustness are implemented in direct response to specific analyses required within the IES program, in LWRS FPOG, in conjunction with industry partners as part of agreements, or occasionally by external users such as universities. Where possible, we designed these development efforts to be as generic and flexible as is practical, such that similar analysis can be performed with minimal tool changes. However, we approach FORCE development as a research software engineering project and are rarely afforded the opportunity to completely develop our tools to cover the full spectrum of problems within their scope.

In the following sections, we describe, by subject, the development improvements made to FORCE tools. For each development activity, we describe the improvements to the software itself; the new capability, accessibility, and/or robustness introduced; new complications discovered as part of the research development; the location of documentation to enable users to access this development; and any follow-on activities introduced as the result of the development.

2.2 FMI/FMU

Within the IES program, INL has been developing models and analysis/optimization tools for the DOE Office of Nuclear Energy. This effort entails enhancement of the capability to exchange and transfer both ROM and existing Python models between different simulation environments. This section covers the recent progress and future directions of the efforts related to FMUs and FMIs.

The FMI standard describes an open format for exporting and importing simulation models via a common data exchange nomenclature. In other words, the FMI standard allows the user to retain the same model while selecting the tools best suited for each type of analysis.

In order to be executed, a FMI is always "shipped" with an FMU. The FMU is the executable that implements the FMI. During the exportation of a FMU, a FMU archive is generated from a systems model. During a FMU import, a systems model is generated from a FMU archive.

Model serialization and usage of FMUs and FMIs in FORCE have progressed throughout the past year. Both new serialization features and a FMU exporter were added to RAVEN.

The new serialization features [7] added to the RAVEN code introduced the ability to export external models as pickled files and input pickled external models. This adds a new way for RAVEN to run existing models. The added input and output from the serialization and the FMU export is summarized in Table 1. The serialization gives RAVEN external models the capability to differentiate between input and output, instead of just specifying variables, which is required for creating the interface for exporting FMU models. In addition, the steps code inside RAVEN was refactored into multiple files so as to be easier to work with. These features paved the way for the FMU export ability.

Table 1. Added IOStep input and output (copied from the RAVEN User Manual).

Input	Output	Resulting Behavior
File	ExternalModel	Load On-Disk Serialized ExternalModel
ExternalModel	File	Serialize ExternalModel to Disk
ROM	File	If type is fmu, Serialize ROM to FMU
ExternalModel	File	If type is fmu, Serialize ExternalModel to FMU

The FMU exporter work was made more robust and merged into RAVEN [8]. Continuing the serialization work, RAVEN now can export both external models and ROMs as FMUs. These FMUs have the FMI inside to specify the inputs and output for using the FMU for co-simulation. This export feature was tested on Macintosh, Windows, and multiple versions of Linux.[a] To support these platforms, the PythonFMU package (placed in the RAVEN contrib directory) had to be extended[b] to support Macintosh (it previously only supported Windows and Linux). A build script was created for building PythonFMU as part of the regular RAVEN compiling. The export code works both when RAVEN is installed with Miniconda, or installed with PIP, a python package manager.

Four new regression tests were added to RAVEN to test both the creation and use of RAVEN-generated FMUs. The first test uses PythonFMU with RAVEN to create an FMU based on an external model. The second test uses PythonFMU with RAVEN to create an FMU based on a RAVEN ROM. The last two tests run scripts without using RAVEN. Instead, one test uses fmpy to test the loading and running of an FMU created from an external model, and the other uses fmpy to test the loading and running of an FMU created from a ROM.

[a] Note that it did not work on CentOS 7 with the default C++ compiler, due to the compiler's missing features.

[b] An issue has been filed with PythonFMU explaining RAVEN's modifications.

Use of RAVEN FMUs with FMIs by accessing tools other than fmpy entails difficulties. For example, getting the FMUComplianceChecker [9] to work for Linux- and Macintosh-generated FMUs required changes to PythonFMU. For Linux, a dlopen of the Python library inside the PythonFMU code was required. For Macnitosh, the linking method had to be changed, and this change allowed FMUComplianceChecker to pass but rendered fmpy unable to use the FMU [10]. A RAVEN-generated FMU was tested in Linux with Dymola. Dymola was unable to use it because Dymola and the FMU were linked with two different C++ libraries, thus causing load failures. However, a Dymola-generated FMU could be used with a RAVEN-generated FMU when both were loaded in fmpy.

The original intent in embracing FMI/FMU as a modeling standard was to foster a simple "plug-and-play" environment that would allow surrogate/reduced-order modelers, transient process modelers, and other digital twin modelers to trivially place various models into a solving system and perform quality analysis. In particular, a key feature of this vision was to avoid any extra coupling efforts on the part of analysts attempting to connect and solve these models together. Despite the ambitious aims of FMI/FMU, however, it is evident that this infrastructure, while promising, does not currently have sufficient maturity to deliver the plug-and-play experience desired under IES. We can continue using FMI/FMU to connect transient models to each other in Modelica, or to connect RAVEN ROMs to each other, but for now notable effort is required by analysts to connect RAVEN ROMs and Modelica models in a coupled solving environment. We will continue to monitor developments in FMI/FMU software and throughout the general modeling community so that we can capitalize on any developments that may lead to a true plug-and-play system.

2.3 TEAL Visualization Improvements

In this section, we discuss three different plots that can now be generated in TEAL to visualize the resultant outputs (e.g., cashflows). These plots are automatically generated thanks to the new developments to TEAL, and are available generically for all TEAL users. Note that, due to time constraints, these have not yet been propagated to HERON, though this could be accomplished with minimal effort. Here, we use a generic power plant with a capital cost and revenue stream from electricity sales as an example. This example case is meant to demonstrate the TEAL mechanics and does not necessarily represent a real physical system. This test case is one of the regression tests included in TEAL's software quality assurance process. The following figures illustrate visualization of economic calculations in TEAL for hypothetical positive and negative cashflows. The negative cashflows could represent initial capital investments and operating expenses for plant components, while the positive cashflows could represent revenues from electricity, hydrogen, or other products. In this example, there are two positive and two negative cashflows for Component 1, and one positive and one negative cashflow for Component 2.

2.3.1 Visualization of Yearly Cashflows by Using Stacked Bar Charts

In this section, we consider yearly cashflows, including inflows and outflows, that are plotted and compared using stacked bar charts. Yearly net cashflow is also visualized to show the difference between year inflows and outflows. The code generates default plots of he annual changes in yearly cashflows, from Project Year 0 to the final year (see Figure 7). Six cashflows, consisting of three inflows and three outflows, are marked in different colors, depending on their values. Red represents cashflows for Component 1; blue represents cashflows for Component 2. The lighter color indicates a higher value.

For Project Year 0, Figure 7 shows an outflow of $1.96 billion for Component 1 (Component1_Negative1) and another outflow of $34 million for Component 2 (Component2_Negative1). Then from Project Year 1 to 16, there are three different inflows (Component2_Positive1, Component1_Positive1, and Component1_Positive2) and one outflow (Component1_Negative2), and the yearly net cashflow remains positive. Component1_Negative2 and Component2_Negative1 stop after Project Year 16, then restart from Project Year 61 to 76. In addition, there are two outflows for Component 2 (Component2_Negative1) in Project Year 40 and 80. Another outflow for the capital cost of the balance of plant (Component1_Negative1) in Project Year 60 makes the yearly net cashflow negative. More details on the annual changes of inflows, outflows, and yearly cashflow can be found in Figure 8 and Figure 9.

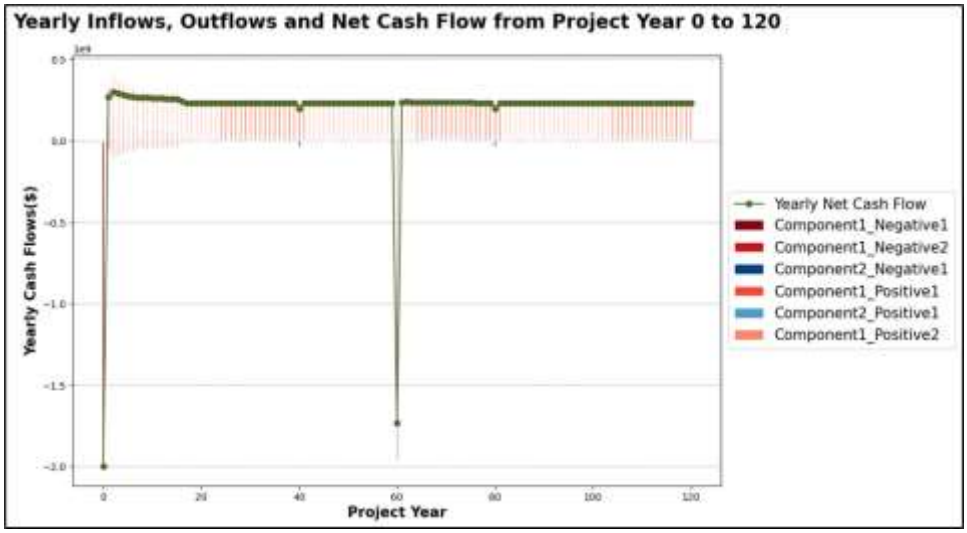

Figure 7. Yearly cashflows from Project Year 0 to 120.

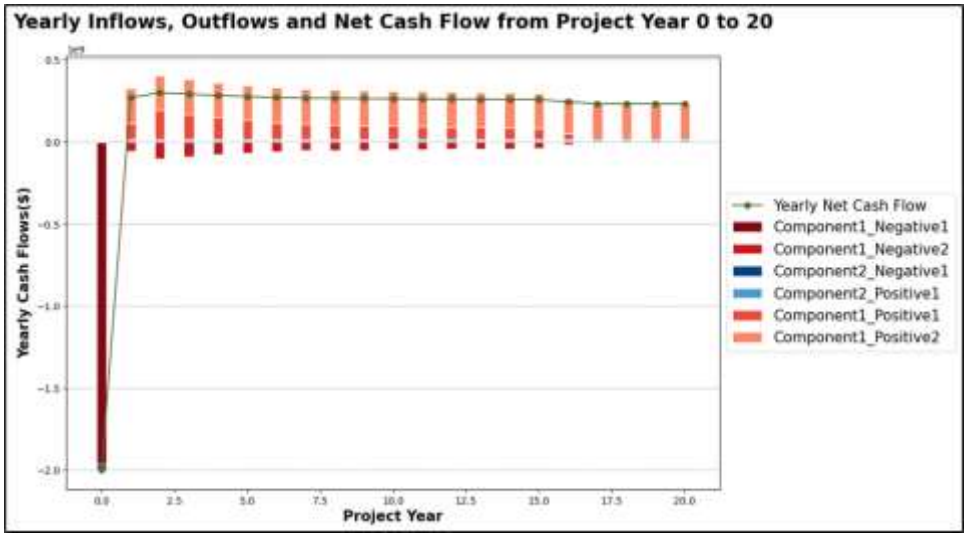

Figure 8. Yearly cashflows from Project Year 0 to 20.

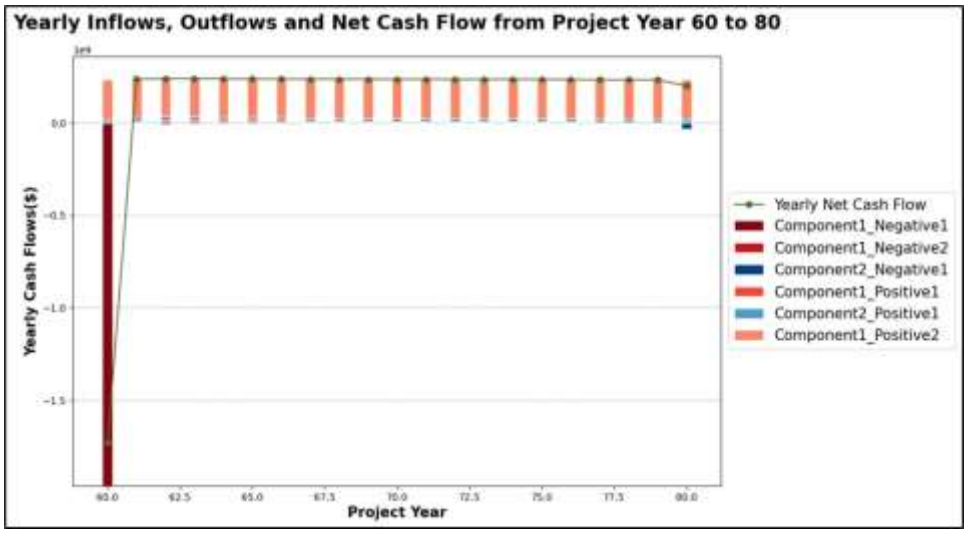

Figure 9. Yearly cashflows from Project Year 60 to 80.

2.3.2 Visualization of Cumulative Cashflows by Using Grouped Bar Charts

In this section, the yearly cashflows, including inflows and outflows, are plotted and compared using grouped bar charts, and are used to calculate and visualize the cumulative net cashflow (without discounting). Discounting is not considered in this report, but will be calculated and used for data reporting and visualization in the future. The code defaults to plotting the cumulative net cashflow from Project Year 0 to the last year (in this case, 120), as shown in Figure 10. The six cashflows, consisting of three inflows and three outflows, are marked in different colors, depending on their component names and values. Red represents cashflows for Component 1; blue represents cashflows for Component 2. A lighter color indicates a higher value.

In Figure 10, the green line that represents the cumulative net cashflows starts from a negative value (about -$2.0 billion) as a result of the outflows of Component 1 and 2 (Component1_Negative1 and Component2_Negative1) in Project Year 0. Then it continually increases until Project Year 60, which involves another build of Component 1 (Component1_Negative1), but again continues to increase after that. These six cashflows are plotted using grouped bars so they can be compared more easily in terms of value. Multiscale y axes are used in this plot: one for yearly cashflows (left side) and one for the cumulative net cashflow (on the right side in green).

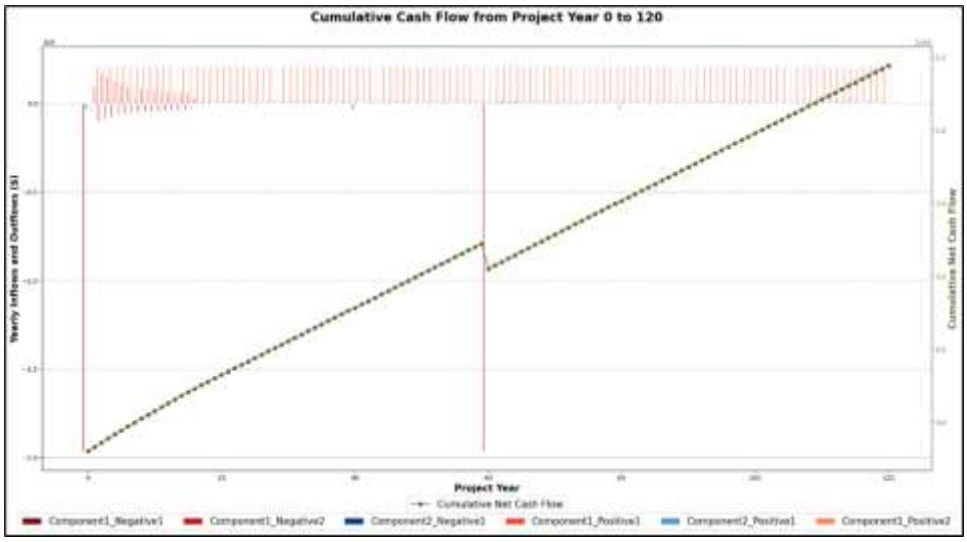

Figure 10. Yearly cashflows and the cumulative net cashflow from Project Year 0 to 120.

One more piece of information obtainable from this plot is the year in which the cumulative net cashflow changes from negative to positive, as shown in Figure 11. The cumulative net cashflow is still negative in Project Year 7 but becomes positive in Project Year 8.

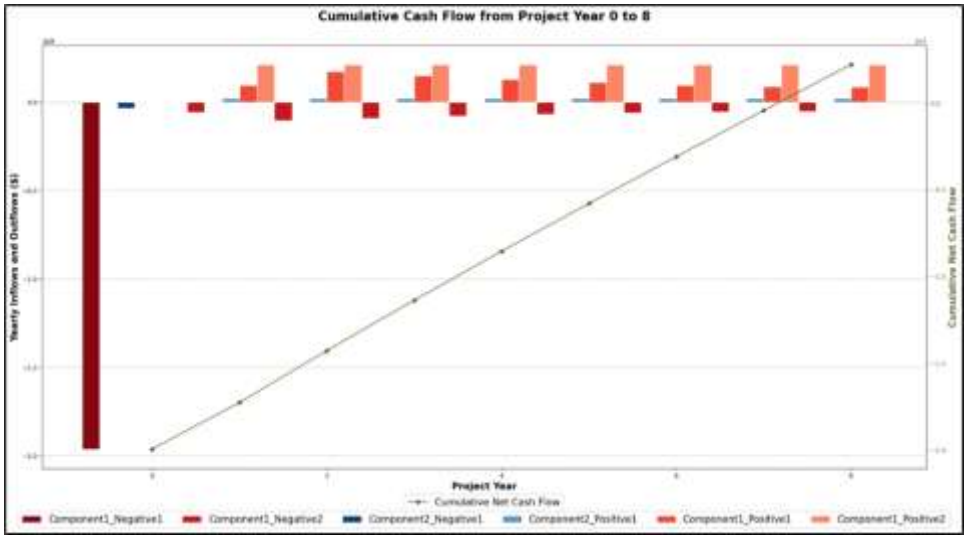

Figure 11. Yearly cashflows and the cumulative net cashflow from Project Year 0 to 8.

2.3.3 Visualization of Cumulative Cashflows by Using Donut Charts

In this section, the cumulative cashflows, including inflows and outflows, are visualized and compared using donut charts. By default, the code plots these cumulative net cashflows from Project Year 0 to the last year (in this case, 120), as shown in Figure 12. The six cashflows, consisting of three inflows and three outflows, are marked in different colors and given different portion sizes, depending on their component names and values. Red represents cashflows for Component 1; blue represents cashflows for Component 2. Lighter colors indicate a higher value.

In Figure 12, the three inflows complete a full donut in red, while the three outflows create only a partial donut. This is because the sum of inflows is higher than the sum of outflows, so the cumulative net cashflow is positive in Project Year 120. The approximate difference between the sum of inflows and the sum of outflows is also shown in the figure.

Figure 13 shows the donut chart from Project Year 0 to 7, with the sum of inflows being lower than the sum of outflows. In this case, three outflows complete a donut in the inner donut, while three inflows only take part of the outside donut. The approximate difference between these two sums is also shown in the figure above the donuts, and means the cost is higher than the income. More examples are given in Figure 14 and Figure 15.

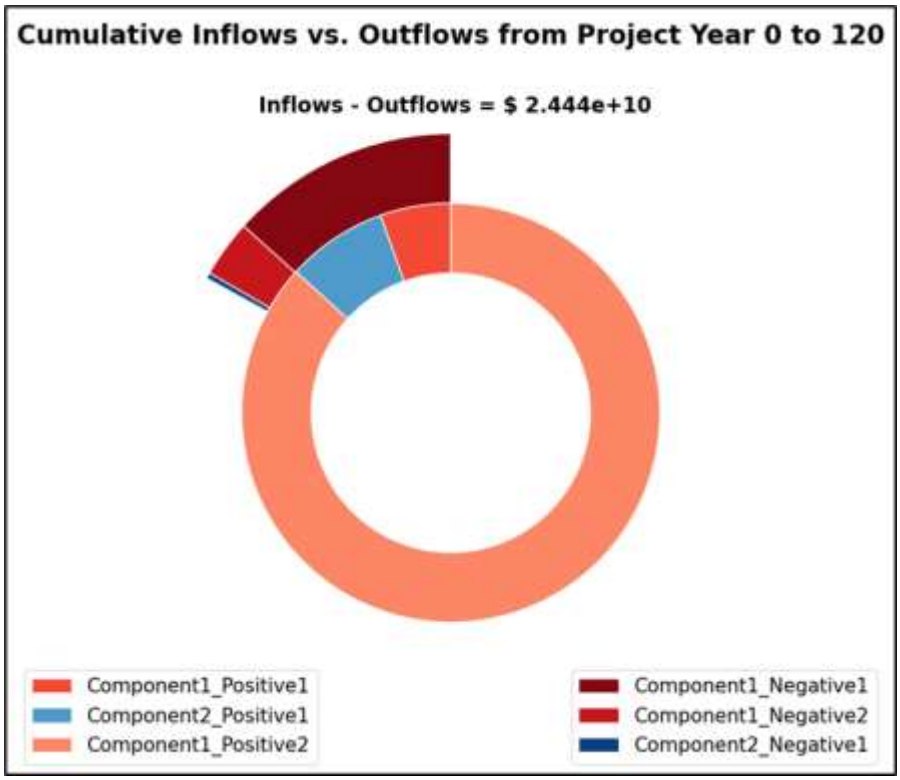

Figure 12. Cumulative cashflows (using donut charts) from Project Year 0 to 120.

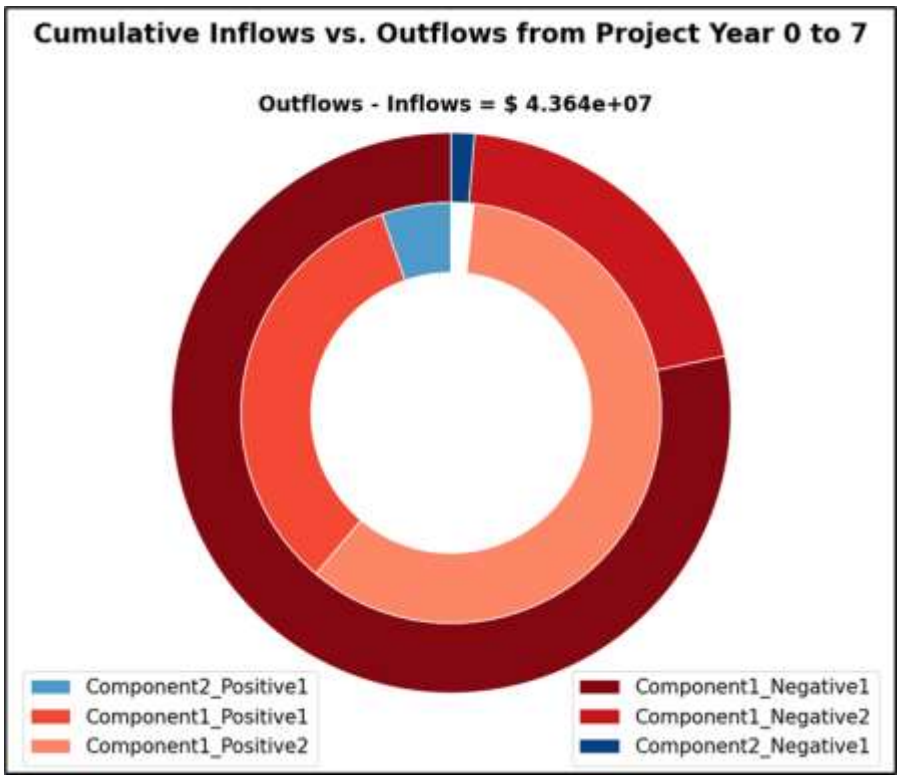

Figure 13. Cumulative cashflows (using donut charts) from Project Year 0 to 7.

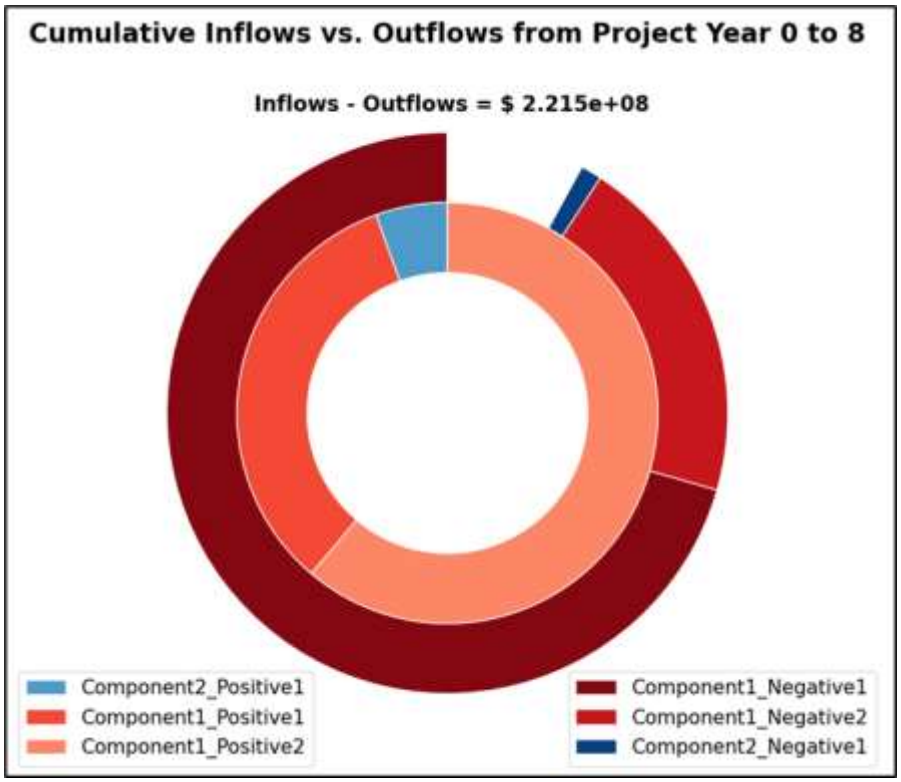

Figure 14. Cumulative cashflows (using donut charts) from Project Year 0 to 8.

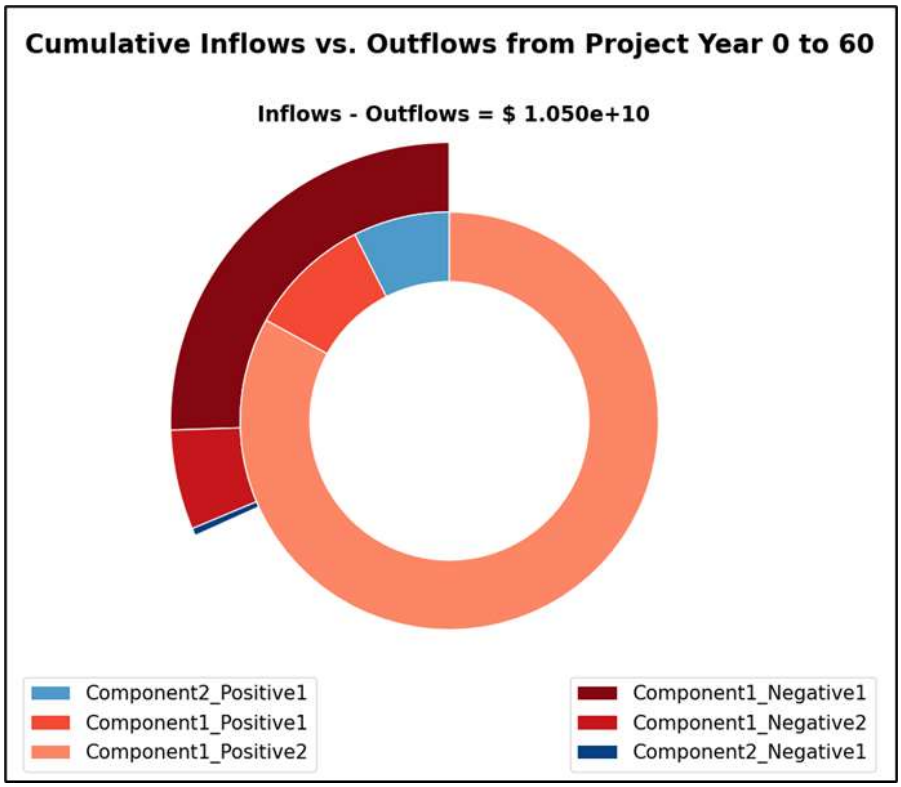

Figure 15. Cumulative cashflows (using donut charts) from Project Year 0 to 60.

2.3.4 Summary of TEAL Visualization Improvements

This work developed the capability for visualization in TEAL via plotting, thus enabling users to better understand and investigate TEAL calculation results. In accordance with the need for and purpose of this visualization, users can select various bar or donut charts to visualize cashflows—including inflows, outflows, and net cashflows—in each year or over a selected time range. The visualization capability is made flexible and user-friendly by introducing several user-defined parameters (e.g., the start and the end project year of interest).

The cashflows can be displayed in different ways for different purposes. Yearly cashflows can be plotted in either stacked or grouped bar charts for different time ranges, depending on whether users plan to compare the sums of inflows and outflows or investigate differences between individual cashflows. Users can decide to plot yearly or cumulative net cashflows: a yearly net cashflow indicates the annual net cashflow changes in each project year; the cumulative net cashflow represents the annual cumulative net cashflow changes over a selected time range. In addition, a donut chart can be plotted if users need to analyze the contribution of each cashflow to the net cashflow for a certain time range. The difference between inflows and outflows is also calculated and displayed using the donut chart.

During development, coloring cashflows according to their originating components was considered. This significantly improves the visual storytelling for TEAL calculations. Furthermore, tools employing TEAL (e.g., FORCE, LOGOS, and HERON) can now be extended to automatically generate these plots to inform users on decisions made during optimization workflows. In addition, establishing within TEAL a framework for plotting will make future visualization enhancements easier to implement.

2.4 Maintenance

RAVEN interacts with different software in order to run. As other software that RAVEN interacts with change, RAVEN must be updated as well.

Various maintenance work was accomplished over the past year. The biggest push has been to improve cluster and parallel running support. Cluster runs are performed on HPC resources and offer much greater parallelizability than a desktop workstation or laptop. While some HERON analyses may be explorable on a desktop machine, in practice, HPC resources are required in order to efficiently perform HERON analyses with any significant degree of precision. Some work was also done to update supporting work with newer versions of Prescient [11,12]. Prescient was previously coupled to RAVEN to serve as a market modeling tool.

For RAVEN itself to be parallelized on clusters, it uses the Ray library.[13] to distribute tasks to different nodes. It was updated to support newer versions of Ray [12]. Tracking down and removing defects and inefficiencies in the cluster implementation has been accomplished. Significant hurdles were encountered by several IES analysts performing HERON calculations. These hurdles were overcome by a combination of library updates and software changes that enabled analysts to perform their calculations.

Because RAVEN now checks for establish_conda_env.sh on the HPC cluster before sourcing it, a warning occurs instead of a crash. RAVEN can now use the same version of Python in both the inner and outer parts of a HERON analysis, instead of just using system Python. This fixes problems when RAVEN is using a different version of Python. RAVEN switched to using ray.get with a timeout instead of ray.wait, since ray.wait did not always seem to get the data results of dispatch optimization [14]. Also, various debugging features were added (only when verbosity = "debug" in RAVEN) to assist in finding problems.

RAVEN no longer uses mpiexec for running inner runs in RAVEN-Runs-RAVEN (unless requested), since it had been interacting badly with Ray sometimes. RAVEN now waits until the Ray servers finish starting, instead of ignoring them (which caused them to become Unix “zombies”). RAVEN now lets Ray find a port, instead of forcing it to a fixed port, which caused problems if two RAVEN runs were being conducted on the same node. Previously, RAVEN had started Ray on the head node, then shut it down and started it up again on the compute node, but now RAVEN just starts it on the compute node. With that, a test of RAVEN-Runs-RAVEN, with both outer and inner runs using Ray, was added [15]. This test will enable consistency regression tests to alert developers when new changes to RAVEN or the supporting libraries cause parallel analysis to cease working as expected. RAVEN was updated so that, by default, it can run on both the INL Sawtooth and Lemhi HPC clusters without having to change configuration files [16].

RAVEN was updated to properly handle various errors and unusual conditions in DistributedMemoryRunner (e.g., when a TaskError was thrown by Ray, or when the job failed to finish). These updates included fixing concurrent access issues that occurred when the optimizer killed jobs. In addition, a test using internal parallel and the optimizer was added to the RAVEN tests [17,18,19].

2.5 FORCE Tool Communication

This section presents the new developments in automating the data communication and vertical integration between two FORCE tools (i.e., HYBRID and HERON). Autoloading of the economic properties of the energy-grid components from HYBRID to HERON is presented in Section 2.5.1. These properties serve as economic drivers or constraints in the analyses performed by HERON. Also, the feasibility of an optimized energy dispatch, as calculated by HERON, was analyzed using HYBRID. Therefore, we automated the process of transferring the optimized energy dispatch from HERON to HYBRID (see Section 2.5.2).

2.5.1 Autoloading the Components' Economic Information from HYBRID to HERON

HERON uses the Extensible Markup Language (XML) for its input file structure. The input XML file comprises three main parts (nodes): (1) the Case node, which includes the general physics and economics information required to create a HERON workflow; (2) the Components node, which includes the technical and economic information of each IES grid component, and (3) the Data Generators node, which processes the data. While all these nodes are created or modified by the user, all information contained in the Components node can be imported from the output of the HYBRID simulations.

In this work, we automated the loading of economic data from the HYBRID text files to the HERON XML input file. Figure 16 shows an example of such data (previously manually transferred), illustrating some of the high-temperature steam electrolysis unit economic information imported from the HYBRID text files [1].

Automation of the data transfer from HYBRID to HERON involved the following assumptions:

- The HYBRID text file(s) data may be incomplete. In other words, the HERON input XML file may require more data before it can be run. Therefore, not only are data autoloaded from HYBRID to HERON, but default values assigned to those variables cannot be extracted from the HYBRID text files but are nonetheless required by HERON. For example, the capital expenditures cashflow type is not found in HYBRID, but its default value is known to be "one time." Furthermore, warning messages (comments) are added to the HERON input XML file and terminal output to notify users of the assigned default values.
- The HYBRID text files may include information irrelevant to creating the HERON input file.
- Units in the HYBRID file may differ from those in the HERON file. Thus, the data auto-transfer also involves transferring the comments, since the units are typically included as comments. As a result, users can review the units (comments) and make any necessary changes.
- The names of the variables in the HYBRID text files are not standardized and can be changed by users. However, in HERON, the names of variables (in the form of nodes and their attributes) are unchangeable.
- A variable in a HYBRID text file may correspond to multiple variables in the HERON input XML file. For example, the "reference size" HYBRID variable (see Figure 16) corresponds to two different nodes in the HERON input XML file.

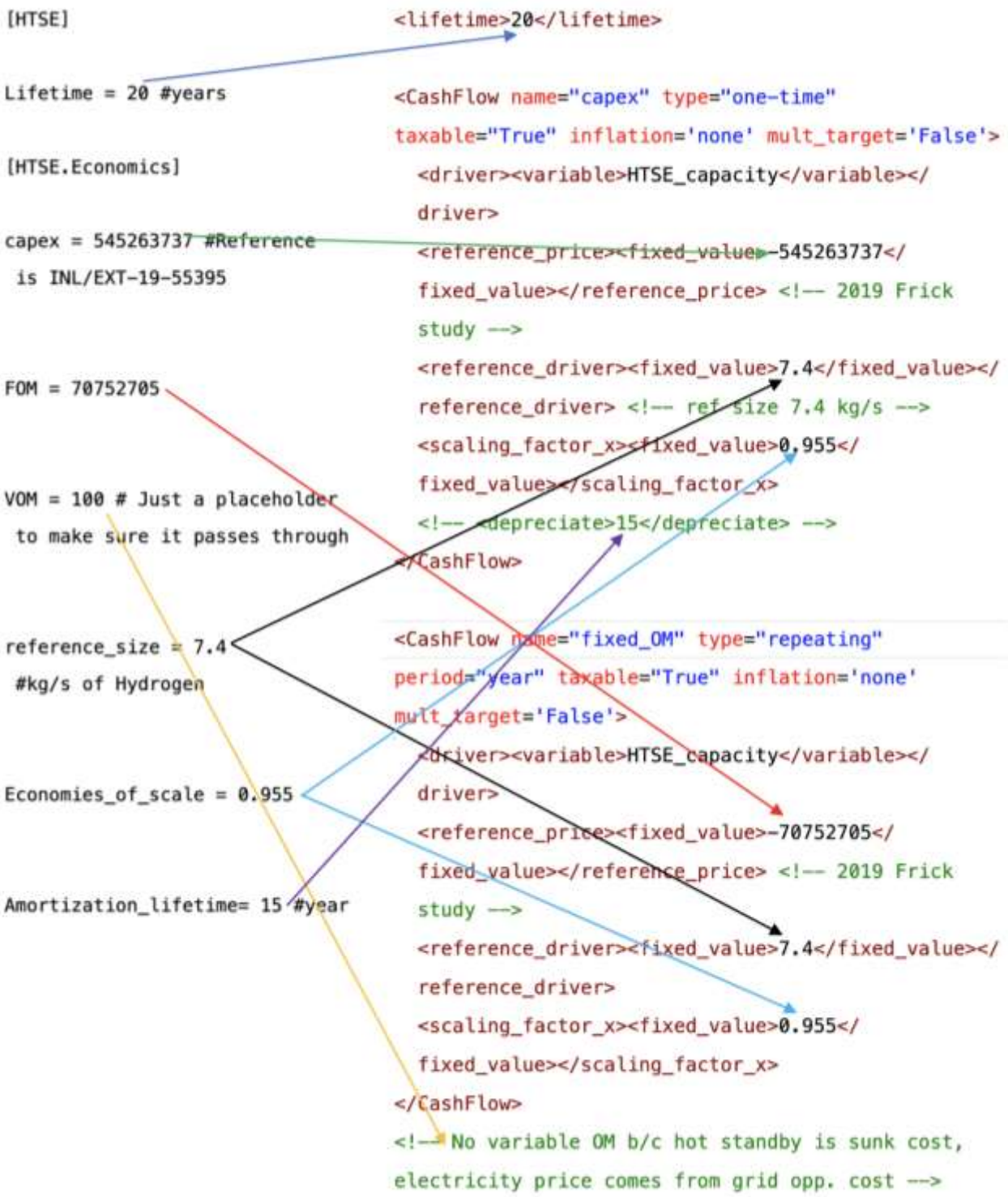

Figure 16. Manual information transfer between a HYBRID text file (left) and the HERON XML file (right).

The process of autoloading data from HYBRID to HERON is represented in Figure 17. The data are transferred via a Python script, whose output is the autoloaded HERON input file.

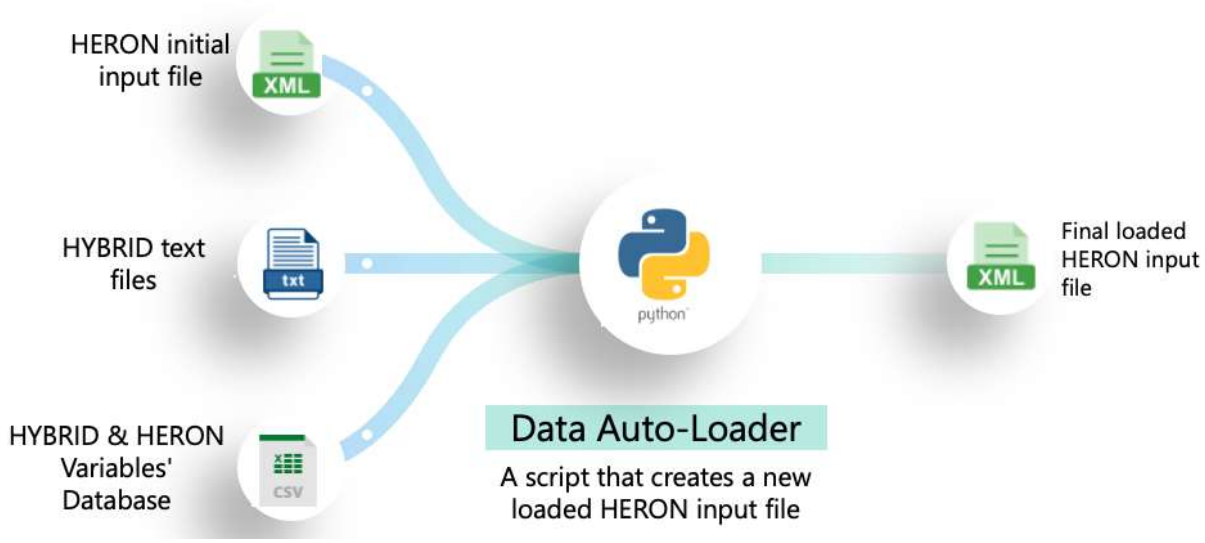

Figure 17. Process of autoloading data from HYBRID to HERON.

The structure of the initial HERON input XML file (on the left in Figure 17) resembles that of the typical HERON input XML file, except that the Economics sub-node found under any Component node does not include any economic data. Instead, users add the location of the HYBRID text file (in Figure 17) from which they wish to import the economic information. Further details on the structure of the HYBRID and HERON input files are explained in the HERON User Manual [20].

Another essential element of the HYBRID-to-HERON data autoloading process is the database (CSV file) that lists the HYBRID variables and their corresponding nodes in the HERON input XML file. Including the HYBRID and HERON variables in this database—instead of the Python autoloader script—is necessary because the names of the HYBRID variables can be changed, and additional variables may be considered in the future. Thus, users will not have to modify the Python code if they plan to add/modify the HYBRID or HERON variables. Instead, simple changes will be made in the database file, which not only includes the names of the HYBRID/HERON variables but also more information on the relations among these variables (see the HERON User Manual [20])

The Python autoloader script autoloads the data from HYBRID to HERON via the following steps:

- Reading the initial HERON input XML file and identifying the Economics sub-node that must be autoloaded from HYBRID under any Component node.
- Reading the HYBRID text file(s) and identifying the name of the variables, values, and corresponding comments.
    - Based on the existing HYBRID variables, additional ones not found in the HYBRID text files may be added, as they are variables required by HERON. This step is necessary to ensure that the HERON input XML file is complete, because the HYBRID files do not always provide all the needed information.
    - Default temporary values are assigned to these additional variables absent in the HYBRID files.
    - Warning messages inform users about the assigned default values, which they must review/modify.
- Creating HERON nodes that correspond to the HYBRID variables, based on the HYBRID-HERON variable database.
- Load each component's Economics node, which contains the new HERON nodes.

This data auto-transfer capability from HYBRID to HERON was tested, and users/analysts can find this test in the HERON GitHub repository [21]. User instructions are documented in the HERON User Manual [20]. Additional details on this data auto-transfer were presented in a previous report [5].

2.5.2 Autoloading the Optimized Energy Dispatch from HERON to HYBRID

An optimized energy dispatch can be calculated via HERON through two steps (i.e., two optimization loops): an outer loop for optimizing the sizes of the grid components/units and an inner loop for calculating the optimal energy dispatch. These two steps (loops) create the optimized dispatch CSV file. To assess the feasibility of the optimized energy dispatch using HYBRID, we must convert the dispatch CSV file into a text file that is compatible with HYBRID. This data exchange between HYBRID and HERON is conducted under the following assumptions:

- The HERON CSV file includes a set of time series or other type of different scenarios (different samples and years) involving the optimized components' variables (dispatches), whereas HYBRID only requires one time series.
- Performing the HYBRID simulations requires obtaining the value of each component's capacity, in addition to the optimized components' production/consumption levels.
- There may be discrepancies between HERON and HYBRID in regard to the variables' names.

Via two steps, the HERON-optimized components' resources (optimized dispatch) CSV file is converted into a text file that is compatible with HYBRID (see Figure 18).

1. Creating the user-input file for the HYBRID user
   This step enables the HYBRID user to change the names, if necessary, of the optimized components' variables and the component capacities. The HYBRID user-input file is created by running a Python code that extracts the optimized dispatch outputs from the dispatches CSV file, and the list of component capacities from the HERON input XML file.

   This step creates the user-input file that the HYBRID user should modify or review before moving to the next step. The user-input file lists the HERON variables, their corresponding HYBRID variables (which the HYBRID user may change), and the location of the *all_variables_file*. The *all_variables_file* is a file that includes all the HYBRID variables. All the capacities and dispatches should be a subset of this file's variables. This *all_variables_file* file helps ensure that the HYBRID capacities and dispatches are identified by HYBRID.

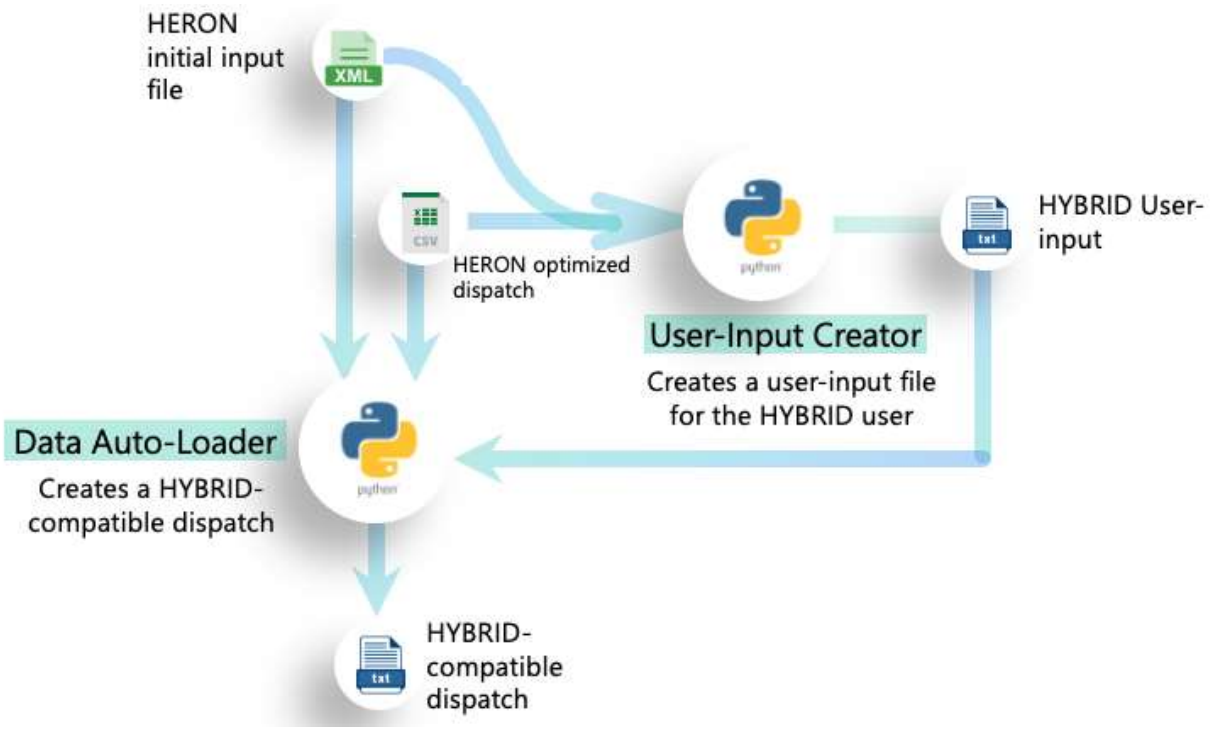

Figure 18. Process of autoloading the optimized components' dispatches from HERON to HYBRID.

2. Creating an optimized dispatches file for HYBRID:
   Using the Python code, a text file containing the optimized dispatches and capacities of the components is created so as to be compatible with HYBRID. The variables' names in the generated (autoloaded) HYBRID text file are borrowed from the user-input file. The Python code extracts the values of the component capacities from the HERON input file and extracts the most challenging scenario from the HERON-optimized dispatch CSV file. We assume the most challenging scenario to be the one (for a specific year and sample) featuring the highest rate of change in any components' resource (variable), because this sudden fast change in production/consumption may not be physically feasible or may violate some operational constraints. Therefore, the most challenging scenario is selected as follows:

   - For each scenario, calculate the rate of change at each time step for each component variable.
   - For each scenario, calculate the maximum rate of change over all the time steps and over all the variables.
   - Select the scenario with the highest maximum rate of change as being the most challenging.

   This capability of autoloading the optimized energy dispatch from HERON to HYBRID was tested, and users/analysts can find this test in the HERON GitHub repository [22]. The user instructions are documented in the HERON User Manual [20].

2.6 DISPATCHES Integration

We added a new class to the HERON source directory to handle the transfer of plant, component, and economic information from a HERON XML input script to the DISPATCHES flowsheets. This class is named HERON Runs DISPATCHES (HERD), and it inherits the MOPED class architecture. A recent addition to HERON, MOPED offers an alternate workflow similar to the DISPATCHES approach: it runs optimization in a single monolithic solve rather than the outer-inner multistage optimization implemented in the HERON-runs-RAVEN method. MOPED conducts its optimization on automatically generated Pyomo models and offers infrastructure to collect component information from the user-supplied HERON XML inputs. HERD now offers a third alternative workflow option within HERON: a monolithic optimization strategy, but using the DISPATCHES flowsheets and workflow with the IDAES Pyomo models, property tables, and other features within its platform. HERD borrows some MOPED methods to leverage existing architecture, and adds new ones to facilitate the gathering of necessary metadata for running the DISPATCHES workflow.

After collecting all component metadata, HERD cross-references against a template dictionary matching the DISPATCHES nuclear case components. The template dictionary is found within HERD and contains high-level replicas of all the DISPATCHES component, including actions (i.e., “produces,” “consumes,” and “stores”). For example, HERD checks that a “pem” component is found in the HERON input script that produces hydrogen and consumes electricity. The DISPATCHES template dictionary also contains metadata for each component cashflow—these take the form of a string with the variable name of the intended cashflow driver in the DISPATCHES nuclear case. When building TEAL cashflows, these strings are used to extract the correct variable from the Pyomo model.

After checking that the HERON XML input contains the same components as the DISPATCHES nuclear case, HERD builds a multiperiod model for the nuclear case, using the DISPATCHES method. The workflow for generating this model is similar to that contained in the “multiperiod_design_pricetaker” Jupyter notebook, depicted in Figure 19. Rather than importing a static LMP signal from a JavaScript Object Notation (JSON) script, as shown in Step 1 of the diagram, HERD loops through all the collected component metadata from the input script and, if a synthetic history is specified, locates the desired ROM and produces a number of sampled histories stated by the user.

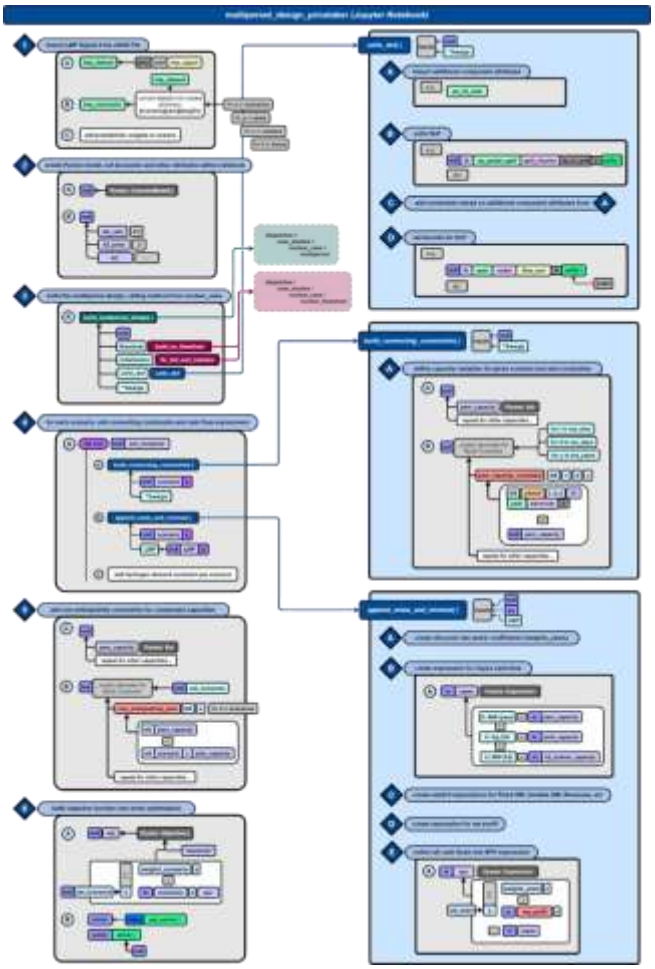

Figure 19. Flow diagram of the Jupyter notebook used for the nuclear case study in DISPATCHES.

To mimic Step 2, HERD imports global metadata from the HERON case and sets the meta data attributes within the Pyomo model object. For easy reference, this object is stored in memory within the HERD class. For Step 3, HERD also calls on the "build_multiperiod_design" method within DISPATCHES, pointing to the available nuclear flowsheet and a method to fix DOFs. The "unfix_dof" method is moved to the HERD class and later used to add/remove model flexibility, as desired by the user. In Step 4, we replace calls to the cost-appending method within the loop through scenarios ("realizations," or "sampled histories" as they are referred to within HERON); we instead use that code block to iteratively create TEAL components and cashflows. This is conducted by cross-referencing the DISPATCHES template dictionary and using the corresponding variables as cashflow drivers, as necessary. We also combine the cost data provided by the user in the HERON input script as the reference cost values within the TEAL cashflows. Step 5 is replicated from the Jupyter notebook to add non-anticipativity constraints (i.e., ensuring that for every scenario, all capacity variables agree). Finally, in Step 6, we replace the NPV expression within the Pyomo objective function with the TEAL-generated NPV Pyomo expression. Users will also have the option of using different metrics within the objective, including IRR or PI.

With the newest additions to the HERD class, HERON can now run the DISPATCHES nuclear-hydrogen case shown in Figure 5 directly from a HERON input file. Rather than generate generic Pyomo models for the different case components, HERON can now utilize the IDAES algebraic models and flowsheets provided within the DISPATCHES repository. HERD can also formulate the optimization scheme via the monolithic multiperiod method by using tools in DISPATCHES. Moreover, HERD now augments DISPATCHES workflows by incorporating the FORCE tool suite. Cashflows are now automatically generated using the TEAL plugin for all user-specified components, costs, and revenues. These are created as Pyomo expressions that are interchangeable within the Pyomo objective function currently implemented by DISPATCHES. Users can now define more complex amortization plans, define individual component lifetimes, and generate multiple metrics for use as objectives within the optimization. HERD also allows for ROM sampling to generate synthetic histories for LMP signals. HERD calls on RAVEN to generate synthetic histories from a user-provided ROM, then runs the multiperiod optimization using these market history "scenarios." Currently, no ROM training capabilities are included in the HERD framework. Users can learn more about the new HERON-DISPATCHES workflow by referring to the HERON User Manual.

There are some limitations to HERD capabilities—to start a simulation, the HERON input file must directly include all components in the DISPATCHES nuclear case. Only the PEM electrolyzer, hydrogen tank, and hydrogen turbine capacities are considered variable. Future customizability will be added to toggle some component capacities from static to variable and vice versa. The simulation also currently uses some hard-coded component values for optimization bounds, conversion factors, etc. This is due to the usage of the static existing nuclear flowsheet in the DISPATCHES repository. Creation of new flowsheets is a matter of future work, with the user enabled to customize more operation-specific component features.

2.6.1 Comparing Results Using DISPATCHES vs. the TEAL-derived Objective Function

The currently available nuclear-case Jupyter notebook for multiperiod price taker analysis in the DISPATCHES repository tests both the deterministic and stochastic usage of LMP signals. The deterministic case is essentially a subset of the stochastic case: it only optimizes over one scenario of LMP signals. The LMP signals in the repository are a collection of five scenarios corresponding to 20 years' worth of data. However, to simplify the calculations in the notebook, the original authors only use two 2-year sets of data for each 20-year scenario. Those two 2-year sets of LMP signal data are repeated over and over again to represent 10 years each (e.g., 2022 data are used as a stand-in for the years 2022–2031, then the 2032 data are repeated in similar fashion). This simplification reduces the number of variables needed for the simulation. Ultimately, the notebook optimization results in a sizable PEM electrolyzer, but based on the static hydrogen price and given LMP signals, negligible hydrogen tanks and turbines are required for the IES. The results are shown in Table 2.

Table 2. Optimization results comparison between the original DISPATCHES notebook and HERD using TEAL-derived cashflows and the objective function.

Capacities	Original Jupyter Notebook Value	% Difference using TEAL
PEM	196.26 MW	-1.08 x 10-9%
H2 Tank	2.0756 x 10-5 kg	-0.439%
H2 Turbine	9.0767 x 10-5 MW	1.31 x 10-4%
NPV (objective)	$1.5968 B	-2.6265%

HERD was also used to generate a DISPATCHES workflow and execute the all-at-once stochastic optimization using TEAL cashflows instead of the generated Pyomo expressions of the original Jupyter notebook. All cashflows were generated using the same global economics parameters as the original notebook (discount rates, taxation, etc.). The same LMP signals from the JSON script were used to calculate the electricity revenue cashflow, taking extra care to replicate the same yearly structure intended by using the 2-year dataset duplication to create the 20-year dataset. In particular, the correct NPV discount rate coefficients needed to be calculated for the 20-year project life, so just claiming a 2-year project life was insufficient (the coefficients could have been pre-calculated as they are in the notebook, but that did not seem an easy path forward using TEAL). A depreciation plan was also implemented to imitate the one used in the notebook—the most similar being a 15-year Modified Accelerated Cost Recovery System schedule. The results in Table 2 show close agreement between HERD and the DISPATCHES notebook optimization.

2.6.2 Future Efforts for HERON-DISPATCHES Integration

Short-term plans for further integration efforts include more robust usage of the ROM sampling. A proof-of-concept demonstration is currently being implemented using a generic ROM that samples loading signals. These loading signals can be loaded into the simulation and be used to replace the previous method of loading LMP signals from a JSON file. While this would not provide meaningful optimization results, it demonstrates the mechanics of using a user-specified number of samples to generate multiple generic histories, and of creating multiple TEAL cashflows using IDAES algebraic models. A ROM specifically trained on LMP data was acquired and will be used as a more suitable replacement for both the JSON file LMP data and ROM-sampled loading signals. Other short-term plans are to replicate the other two IES cases (renewables and fossil fuel), using the multiperiod model builder in DISPATCHES.

Longer term plans for HERON-DISPATCHES integration (for FY-2023) involve using HERD to automatically generate flowsheets for use in a multiperiod model. The user would be limited to the list of available unit models in the DISPATCHES and IDAES platform but could model and simulate IES configurations beyond the three available cases. An enhanced strategy is needed to determine the software architecture for this workflow (e.g., whether the flowsheets would live in computer memory or be written to scripts or notebooks). There may also be some limitations on which unit models can be coupled.

2.7 HERON Improvements

As mentioned in Section 1.10, the HERON development team focused on several software improvements during FY-22. The focus of the development was to make HERON more accessible, capable, and reliable. Many of the improvements were pre-identified as part of the IES programmatic planning, but some were implemented in ad-hoc fashion. These improvements typically arose from a user request submitted via our public GitHub repository, or from the programmatic requirements of different use cases. The following subsections detail a handful of the most significant changes made to HERON during FY-22.

2.7.1 Static Histories

One of the most significant changes to HERON during FY-22 was the ability to use a CSV file containing a time-dependent signal as the primary form of market data for a simulation. Previously, HERON required its users to train a synthetic history ROM. This could be cumbersome for analysts just looking to verify simulation behavior in the early stages of research.

Now, instead of training a ROM, users can provide a single CSV file containing the information pertinent to their simulation. Since CSV is a common file format used by data vendors to distribute their data, this new addition to HERON will allow users to utilize data directly from the source. This reduces the time needed to prototype a workflow, and increases HERON's accessibility.

While this new feature reduces initial startup costs, it should be noted that only using a single realization of data reduces the robustness of HERON's results. Without considering the uncertainty of data within a given time interval, the optimizer only has one sample to work with. Thus, users are advised to eventually train a ROM in order to get the full robustness of a typical HERON workflow. Future work in this area will allow users to submit multiple realizations in CSV form, and this is expected to significantly reduce this limitation.

This new feature is well-documented within the HERON User Manual. Also, as more use-cases are added to the FORCE repository, users should be able to see examples of this feature in action. It is also planned that future workshops and presentations will leverage this feature when introducing users to HERON.

2.7.2 Monolithic Optimization Workflow

A new workflow option was added to HERON during FY-22. This new workflow, called MOPED, allows users to completely change the way HERON solves fs. As mentioned earlier, HERON operates under a bi-level optimization strategy that can be visualized as two coupled loops. This is the default workflow because it is less memory-intensive and can handle large-scale simulations at the cost of long runtimes on a HPC cluster.

MOPED, however, is a great workflow choice when the modeled energy system is small and simple. Simulations using MOPED are expected to run in a fraction of the time of a typical HERON workflow. Since both workflows produce the same answer, MOPED can be strategically used to test quick changes to a system. In the future, MOPED could be used to completely replace the current default workflow, but would require additional developments to reach feature parity.

MOPED is limited to techno-economic analyses in which the dispatch and capacity selection agents are cooperative. In other words, MOPED cannot solve analyses for which maximizing the dispatch value reduces the total NPV value. Possible scenarios include deregulated markets, direct competition, and agent-based dispatch. Additionally, MOPED is limited in terms of acceptable inputs, which currently include but are not limited to:

- Capacities for VRE (wind/solar)
- Reference prices for synthetic histories
- Components that consume one resource to produce another
- Storage components
- Custom functions for prices, capacities, demand, etc.

Future follow-on activities will involve adding features from this list and utilizing the workflow across several IES use cases.

2.7.3 New Optimization Features

Several new optimization features were added to HERON during FY-22. These features provide flexibility to users building out their analyses. Some of the most profound new features include:

- Added round-trip efficiency (RTE) to storage components
- Added value at risk as an economic optimization target

- Added arbitrary optimization variables via HERON input
- Added the "persistence" option to capacity optimization
- Added capacity optimization convergence criteria.

Adding RTE to storage components allows users to consider battery and storage efficiency during dispatch optimization. RTE is specified as a floating-point value between zero and one, and can be declared in the HERON input file. Previously, it was not possible to specify an RTE, and HERON assumed that storage technologies had perfect 1:1 efficiency. It was discovered, after the addition of RTE, that the optimizer would sometimes leverage the inefficiency of a component to burn off excess energy. While this typically has an inconsequential impact on results, it should be noted as an expected optimization behavior.

Value at risk is a new economic optimization target described as a risk-weighted NPV metric. This metric shows a lower bound on financial loss when encountering an event occurring two standard deviations away from what is normally expected. Also, during the work on adding the value-at-risk capability, we were able to create an optimization metric harness that will eventually allow users to conveniently hot-swap optimization targets. Some of the metrics expected to come from future work will be to optimize the levelized cost of storage.

In similar fashion to the newly added metrics harness, the capability to optimize based on arbitrary variables was added to HERON. Now users can optimize variables not necessarily associated with specific components. This feature gives users added flexibility and lets them construct complex systems for simulation. Along with this feature, optimization persistence and convergence criteria were added as acceptable HERON inputs.

2.7.4 Windows Support

As more analysts adopt FORCE into their workflow, we have seen an increase in Windows-related issues and requests. Currently, FORCE is primarily developed on and for the Macintosh and Linux platforms, but with the increase in Windows users, changes were required to ensure the validity of FORCE on all common computing platforms. Now HERON's documentation can be fully compiled and viewed on Windows computers. Furthermore, the regression tests for Windows are now running in HERON's testing harness. Future work will instantiate a new testing server to run all tests specific to Windows.

2.7.5 Library Improvements and General Maintenance

Finally, many of the small changes made to HERON during FY-22 could be categorized as general maintenance. These changes were often required in the development of use cases and were implemented in ad-hoc fashion. While many of the changes do not have an immediate benefit for end users, they contribute to the overall quality of the codebase and pave the way for increased efficiency in future work.

Update Pyomo Dependency – During FY-22, the development team updated Pyomo, an open-source optimization library, from version 5.7.0 to version 6.1.0. This update allows for the use of grey-box models in HERON, enabling users to couple a theoretical structure with data to create a model. Additionally, Pyomo version 6.1.0 offers improved handling of stochastic optimization thanks to faster runtimes and better formulation syntax.

RAVEN as a Library – Recent changes to RAVEN have made it possible to install RAVEN as a library instead of a codebase. This makes RAVEN more versatile in its ability to integrate with scripts and other codes. HERON was updated to leverage RAVEN as a library, thus removing the requirement to co-locate software—and taking another step toward software-as-a-library installation.

Improved User Options – More user-focused options were added to the HERON input file during FY-22. These options include setting varying levels of verbosity for HERON outputs. We found that some HERON runs were generating several gigabytes of output data that users may not require for production runs. Allowing users to set the verbosity of their runs gives them flexibility during debugging and production runs of HERON. Additionally, more memory usage options were added to HERON. The way HERON transfers information between runs can drastically change its runtime. One memory option allows for better introspection but increases the runtime overall. The other option provides quicker runtimes but reduces the accessibility of the information.

Improved Debugging Abilities – Small changes implemented in HERON's debugging mode make life a little easier for users. The automated plots detailing the dispatch of different system resources have been improved through a more accessible color palette and better use of white space. These plots can also be customized to only show for a specific number of years and realizations.

3. CONCLUSIONS

In this work, we enumerated the improvements made through several activities to support research and analysis in the IES program, as well as in partner programs, using the FORCE tool suite. For a demonstration of these tools in analysis work, we invite the reader to observe the various analyses performed—particularly in the IES program, but also those set to be performed in the near future. We have, however, demonstrated the new capabilities, accessibilities, and robustness introduced by these development efforts, in some cases without the full context of techno-economic analysis. We continue to watch for capabilities that will be needed to perform critical analysis, opportunities to increase the tool suite's accessibility for energy analysts at all levels, and gaps in robustness that can be shored up to improve tool performance.

Improvements to the FORCE tools are made in accordance with stringent nuclear quality assurance (NQA-1) software quality assurance standards. New code additions undergo scrutiny and review, both by automated processes (e.g., regression testing) to ensure consistent performance, and by quality control through other software developers. This process of design, development, review, and inclusion ensures that many tools exist to maintain a high standard of quality for the FORCE tool suite.

3.1 Summary of Improvements

The individual development activities carried out this year were grouped by category and explained in detail above. Additional details on the developed tools are available on their respective FORCE tool repository pages. All FORCE tools are currently free and exist as open-source software accessible in publicly available repositories:

- FORCE: https://github.com/idaholab/FORCE
- HERON: https://github.com/idaholab/HERON
- HYBRID: https://github.com/idaholab/HYBRID
- RAVEN: https://github.com/idaholab/raven
- FARM: https://github.com/Argonne-National-Laboratory/FARM
- TEAL: https://github.com/idaholab/TEAL

For convenience, we will now briefly summarize the development categories.

3.2 FMI/FMU

Research continued on the viability and interoperability of the FMI/FMU standard for Modelica models, as well as for other models such as RAVEN ROMs. While we were able to demonstrate successful generation of RAVEN ROMs with FMI/FMU, they were not "plug-and-play" with Modelica FMI/FMU models, and required significant effort to couple. This led to the conclusion that, though FMI/FMU may offer the promise of an extraordinarily flexible model ecosystem, that promise has not yet been realized by this technology. In our estimation, real-time optimization and similar applications will be better served by coupling ROMs rather than mixing and matching FMI/FMUs from a variety of sources.

3.3 TEAL Visualization Improvements

To increase cashflow visibility and explainability in economic modeling, automated plots were added to TEAL to show year-by-year individual and cumulative data, in addition to donut charts representing the relative impact of each cashflow in a given problem. While these visualizations have not yet been propagated to HERON, following up with this activity will dramatically reduce the manual effort analysts must invest to understand and debug why particular decisions are made by HERON, and the relative impact of various costs and revenues.

3.4 Parallel Communication and Operations Maintenance

Significant effort was made to ensure that appropriate computational resources were leverageable by FORCE tools, especially HERON. Ongoing changes to computing infrastructure and software required notable changes to RAVEN, and thus to HERON as well. As a result, HERON can be run on high-performance computers at INL, with more reliability than previously seen. We also anticipate additional maintenance on parallel computing to be vastly reduced thanks to this effort.

3.5 FORCE Tool Intercommunication

FORCE tools have previously been largely independent. To reduce analyst error and increase efficiency for analysts, efforts were made to establish automation pathways between HYBRID and HERON. Transplanting component operation information from HYBRID to HERON was the first demonstration, followed by using HERON-optimized dispatch results as HYBRID set points for additional analysis. While these communication pathways have not yet been established to their full potential, this initial demonstration paves the way for easily increasing future communication options.

3.6 DISPATCHES Integration

Ongoing work by the DISPATCHES team has yielded optimization algorithms similar to those used by HERON's RAVEN workflows, but with unique strengths and weaknesses. To allow users access to the best tools for solving each problem, initial effort was put forth to demonstrate how HERON could generate and solve DISPATCHES workflows. This led to a toggleable option in the HERON input to indicate which solver system (i.e., RAVEN-running-RAVEN or DISPATCHES) should be employed. Due to the late start of this work and the level of funding, we only initially demonstrated HERON coupling to one particular DISPATCHES workflow, representing one particular IES configuration. With additional effort, this could be extended to more general cases.

3.7 HERON Improvements

A great number of small usability improvements in HERON were implemented as non-developer analysts began using the tool to optimize IES configurations. These led to additional capabilities and flexibility, in addition to increased accessibility for HERON. Further, several targeted developments were completed, particularly the ability to run HERON with static instead of synthetic histories. While running with static histories fundamentally solves a different problem than using synthetic histories, and may lead to misleading conclusions, as a debugging tool, static histories remove uncertainty from the problem and allow analysts to check the system behavior.

3.8 Roadmap and Path Forward

As is always the case, especially for research software, there is a plethora of development opportunities in regard to capability improvements, accessibility features, and robustness.

The robustness of the FORCE tool suite is particularly lacking when it comes to automated testing and deployment. Currently, HERON and HYBRID each only use a single Linux-based operating system to test software changes, leading to Windows users frequently finding that the code stops working after changes are deployed. With some minimal effort, testing could be expanded to include a handful of operating systems (e.g., Windows, Macintosh, and Linux) in order to ensure compatibility.

Similarly, FORCE tools currently expect users to compile user manuals and guides to stay up to date with changes merged by developers. However, most users lack the software basis necessary to compile these documents, and rely on old static versions that are rarely updated. Automating the build and deployment of user documentation, based on software updates, would considerably decrease errors in user inputs and provide accurate resources for using FORCE tools.

In the realm of accessibility, there has frequently been a desire for GUIs for all the FORCE workflows. Currently, FORCE tools use command-line text-based inputs and code operation. This approach is familiar to the older generation of nuclear engineers and developers working on early editions of the software, but is unfamiliar to most energy analysts and entails a difficult learning curve. This difficulty is observed at national laboratories but is even more pronounced in industry and universities. While creating a full GUI for all workflows is a challenging and complex task, there are levels of improvement that could be made sequentially, leading to a full application-type experience. Activities such as integrated text editing environments would offer users suggestions on input file entries and help locate errors before running the code, tremendously accelerating what is currently a slow iterative process. Environments such as the Nuclear Energy Advanced Modeling and Simulation (NEAMS) Workbench have demonstrated success in this area, even to the point of templating input files and guiding users through input generation. This could provide a steppingstone toward an application-style experience.

A significant number of capabilities are missing from FORCE tools, thus preventing it from being an effective, well-rounded suite for IES analysis. For example, multi-time resolution analysis would enable HERON and future real-time optimization to consider multiple scales of decision making, such as seasonal, day-ahead hourly, real-time 5-minute, and sub-minute decision making. This would allow for such technology as seasonal storage—an elusive problem for traditional energy analysis tools—to be captured alongside fast-moving energy markets. While several multi-resolution solution methodologies exist, they would need to be adapted to function with reasonable computational tractability in energy market applications.

Several smaller-effort improvements to HERON could also be made, requiring minimal effort but achieving great impact. First, cost targets are a topic of frequent discussion in IES and advanced NPP consideration. While HERON currently only optimizes for statistical NPV or NPV value at risk, it would be straightforward to recast the problem as a cost target optimizer and determine the price at which various technologies could compete in a given market. This might be especially useful in thermal energy storage analysis, where currently predicted costs prevent arbitrage from being sufficiently competitive. In such analysis, we frequently see analysts manually searching for the capital costs that lead to a break-even profit. We could automate this calculation easily.

On a similar note, HERON currently captures uncertainty via weather, demand, and price data. However, significant uncertainty exists in costing data, especially for technologies that have not yet entered the market. With minimal effort, HERON could accept those uncertainties and propagate them through the statistical dispatch analysis, giving a more holistic view of the cost-benefit analysis of a particular grid portfolio. Because RAVEN—on which HERON is built—already effectively deals with uncertainty, this extension should be straightforward to both implement and demonstrate.

Finally, but perhaps most importantly, there is significant opportunity to improve both the process and quality of synthetic history training for HERON and real-time operations. Existing methods work well for a small subset of signals, but especially for inertial studies such as those including storage and ramping components. Fourier-ARMA has remarkable shortcomings that could lead to inaccurate conclusions. Currently, the selection and training of synthetic histories is as burdensome in terms of analyst hours as the actual economic viability analysis, but only because we have not researched the algorithms and developed the tools to help analysts efficiently train these models. With time and effort, the complication and inaccuracies of training synthetic history models could be reduced by orders of magnitude, allowing time to be spent on other crucial parts of the analysis.

4. REFERENCES

1. Frick, K., et al. 2019. “Evaluation of Hydrogen Production Feasibility for a Light Water Reactor in the Midwest.” INL/EXT-19-55395, Revision1, Idaho National Laboratory, Idaho Falls, ID. https://inldigitallibrary.inl.gov/sites/sti/sti/Sort_18785.pdf.
2. Epiney, A. et al. 2019. “Economic Assessment of Nuclear Hybrid Energy Systems: Nuclear-Renewable-Water Integration in Arizona.” INL/CON-19-52394, Idaho National Laboratory, Idaho Falls, ID. https://inldigitallibrary.inl.gov/sites/sti/sti/Sort_16057.pdf.
3. Epiney, A., et al. 2020. “Economic analysis of a nuclear hybrid energy system in a stochastic environment including wind turbines in an electricity grid.” *Applied Energy* 260:114227. doi:10.1016/j.apenergy.2019.114227.
4. Gunter, D., et al. 2021. “Design, Integration and Synthesis Platform to Advance Tightly Coupled Hybrid Energy Systems (DISPATCHES) v0.1.0.” doi:10.11578/dc.20211028.12. https://github.com/gmlc-dispatches/dispatches.
5. Yodwong, B., et al. 2020. “Exchange Membrane Electrolyzer Modeling for Power Electronics Control: A Short Review.” *C Journal of Carbon Research* 6(2):29. doi:10.3390/c6020029.
6. Taamallah, S., et al. 2015. “Fuel flexibility, stability and emissions in premixed hydrogen-rich gas turbine combustion: Technology, fundamentals, and numerical simulations.” *Applied Energy* 154:1020-1047. doi:10.1016/j.apenergy.2015.04.044.
7. Cogliati, J., and Alfonsi, A. “Cogljj/serialization exporter #1608.” Github. Revised August 10, 2021. https://github.com/idaholab/raven/pull/1608.
8. Alfonsi, A., and Cogliati, J. “Exporter for FMI/FMU #1481.” Github. Revised February 14, 2022. https://github.com/idaholab/raven/pull/1481.

9. Modelica Tools. “FMUComplianceChecker.” Revised May 18, 2020. https://github.com/modelica-tools/FMUComplianceChecker.
10. Idaho National Laboratory. “Fmu work #1838.” Github. Accessed September 2022. https://github.com/idaholab/raven/pull/1838.
11. Idaho National Laboratory. “Support current Prescient version #1744.” Github. Revised January 17, 2022. https://github.com/idaholab/raven/pull/1744 and
12. Idaho National Laboratory. “Combined fixed #1787.” Github. Revised March 14, 2022. https://github.com/idaholab/raven/pull/1787.
13. The Ray Team. “Ray 2.0.0.” Accessed September 2022. https://docs.ray.io/en/latest/index.html.
14. Idaho National Laboratory. “Fix various cluster issues #1807.” Github. Revised April 14, 2022. https://github.com/idaholab/raven/pull/1807.
15. Idaho National Laboratory. “Parallel improvements #1825.” Github. Revised May 19, 2022. https://github.com/idaholab/raven/pull/1825.
16. Idaho National Laboratory. “Lemhi stuff #1850.” Github. Revised June 13, 2022. https://github.com/idaholab/raven/pull/1850.
17. Idaho National Laboratory. “Catch RayTaskError and add debug info #1852.” Github. Revised June 14, 2022. https://github.com/idaholab/raven/pull/1852.
18. Idaho National Laboratory. “Combined lock #1899.” Github. Revised July 20, 2022. https://github.com/idaholab/raven/pull/1899.
19. Idaho National Laboratory. “More parallel work.” Github. Revised August 1, 2022. https://github.com/idaholab/raven/pull/1902
20. Talbot, P., et al. 2022. “HERON User Guide.” INL/EXT-20-58976, Revision 1, GDE-939, Idaho National Laboratory, Idaho Falls, ID.
21. Hanna, B. 2022. “HERON/tests/integration_tests/mechanics/optimizedDispatch2Hybrid/.” Github. https://github.com/idaholab/HERON/tree/devel/tests/integration_tests/mechanics/optimizedDispatch2Hybrid.
22. Hanna, B. 2022a. “HERON/tests/integration_tests/mechanics/hybrid_load/.” Github. Accessed September 2022. https://github.com/idaholab/HERON/tree/devel/tests/integration_tests/mechanics/hybrid_load.
23. Saeed, R., Hanna, B., and Talbot, P. 2022. “FORCE Development Status Update: Vertical Integration and Benchmarking of System Dynamics.” INL/RPT-22-02333, Idaho National Laboratory, Idaho Falls, ID.