INL/EXT-19-56933

Light Water Reactor Sustainability Program

HERON as a Tool for LWR Market Interaction in a Deregulated Market

December 2019

U.S. Department of Energy

Office of Nuclear Energy

INL/EXT-19-56933

HERON as a Tool for LWR Market Interaction in a Deregulated Market

Paul W. Talbot
Abhinav Gairola
Prerna Prateek
Andrea Alfonsi
Cristian Rabiti
Richard D. Boardman

December 2019

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy

ABSTRACT

This document reports the development of HERON (Holistic Energy Resource Optimization Network), a newly-developed RAVEN (Risk Analysis Virtual ENvironment) plugin for grid and capacity optimization, to technoeconomic analysis in a deregulated market. A short description of the HERON plugin is provided, and the release process is described. HERON as a plugin enables RAVEN to perform stochastic technoeconomic analysis of grid-energy systems in a generic approach. The primary function of HERON is to generate the complex RAVEN workflows necessary to optimize component capacities under stochastic systems. HERON is capable of analyzing systems with complex components transferring a variety of commodities, including production components and varied markets. HERON is capable of optimizing high-resolution dispatch for such systems and guiding stochastic optimization algorithms in RAVEN for finding optimal component capacities. In particular, this document demonstrates the application of HERON to systems with deregulated markets.

CONTENTS

FIGURES

TABLES

ACRONYMS

ARMA	Auto Regressive Moving Average
APS	Arizona Public Service
CAPEX	CAPital EXpenditure
CSV	Comma-Separated Values
EG	Electrical Grid
HERON	Holistic Energy Resource Optimization Network
HES	Hybrid Energy Systems
HPC	High-Performance Computing
IES	Integrated Energy Systems
IRR	Internal Rate of Return
LCOE	Levelized Cost Of Electricity
LMP	Locational Marginal Pricing
LWR	Light Water Reactor
MACRS	Modified Accelerated Cost Recovery System
MWh	Megawatt Hour
N-R HES	Nuclear-Renewable Hybrid Energy Systems
NPP	Nuclear Power Plant
NPV	Net Present Value
O&M	Operations and Maintenance
PWR	Pressurized Water Reactor
RAVEN	Risk Analysis Virtual ENvironment
VRE	Variable Renewable Energy source
XML	eXtensible Markup Language

HERON as a Tool for LWR Market Interaction in a Deregulated Market

1. INTRODUCTION

This document reports the development of HERON (Holistic Energy Resource Optimization Network), a newly-developed RAVEN (Risk Analysis Virtual ENvironment) plugin for grid and capacity optimization, to technoeconomic analysis in a deregulated market. Significant efforts have been made in recent years [1-7, 9-12, 16] to construct an analysis framework using RAVEN for systems of energy producers and consumers in a method that allows statistically-meaningful decisions to be made with regards to the economic viability of energy grid portfolios and configurations. The results of this effort have shown value both in a theoretical sense [1, 2, 4, 5, 7, 9-12] as well as for specific applications when paired with industrial partners in the energy space [3, 16]. However, the development and maintenance of these RAVEN workflows has been burdensome, and often involves complicated copy-and-modify operations that significantly slow the user experience. Further, a custom optimization algorithm was required for each configuration of the components in a grid system, further exacerbating the high bar of entry for using these workflows.

To mitigate these difficulties, HERON has been developed as a plugin of RAVEN. First, with RAVEN's recent enhancement to allow automated Templating, HERON can now support inputs that are more intuitive for grid energy system analysts and generate the complex RAVEN workflows required in an automated fashion. Second, in an attempt to solve dispatch optimization for generic systems, HERON provides an algorithm built to handle complex and flexible grid configurations with nonlinear and unpredictable components. These two features are the cornerstones for HERON's software design and requirements.

The development and maintenance of HERON requires clear plans for software design, capabilities, and quality assurance. Inheriting from the RAVEN software quality assurance documentation, the infrastructure as well as maintenance and development of HERON are documented to provide this clear vision.

To demonstrate the efficacy of this new plugin, HERON is applied to a grid configuration containing an electricity producer, a hydrogen producer, hydrogen storage, an electricity market, and a hydrogen market. By economically choosing the dispatch of these components given a stochastic price history, HERON demonstrates the ability to optimize dispatch. By choosing the optimal size of each of these components, HERON demonstrates the ability to create the complex workflows required for stochastic technoeconomic analysis.

The remainder of this document proceeds as follows. Section 2 describes the use cases, design, and capabilities of HERON. Section 3 defines the software infrastructure for HERON, including its interactions with RAVEN and internal software structure. Section 4 describes the software quality assurance practices in place to assure HERON's functional development and defines existing integration tests. Section 5 defines and demonstrates an example deregulated market system using HERON. Finally, section 6 summarizes the results of this document and outlines work moving forward.

In two appendixes we append the software quality assurance (SQA) documentation, read from their PDF form. The first appendix is the combined design, requirements, and traceability matrix. The second appendix is the configuration items list.

2. HERON DESCRIPTION

HERON is a generic software plugin of RAVEN to perform stochastic technoeconomic analysis of grid energy-resource systems with economic drivers. The development targets analysis of electricity and secondary product generation and consumption in regional balancing areas, including flexibility to include arbitrary resources as well as arbitrary resource consumers and producers. HERON is developed to drive optimization via economic drivers such as system cost minimization, profitability, and net present value (NPV) maximization.

As a plugin of RAVEN, HERON provides two primary functions: the automatic generation of RAVEN workflows, and models for optimizing high-resolution dispatch of arbitrary systems including resources, resource consumers, and resource producers. HERON leverages the synthetic history training and generation tools, sampling workflows, code Application Programming Interfaces (API), and optimization schemes.

2.1 Use Cases and Design

HERON is designed to analyze systems of resources, resource producers, and resource consumers in an economic fashion, using characteristic technical features as constraints. Resources can be commodities such as electricity, natural gas, or hydrogen, or more abstract concepts such as wind speed, solar irradiation, or pressure. Resource producers are units that produce one or more resources, such as a nuclear power plant producing steam and electricity or fleet of solar panels producing heat and electricity. Resource consumers can be producers of other resources, such as a natural gas power plant consuming gas to produce electricity, or a sink such as a residential sector consuming power or a hydrogen demand market buying hydrogen. Further, HERON models storage units that can act sometimes as consumers and sometimes as producers, depending on the system configuration and status. HERON performs analysis of systems at both an intra-year dispatch resolution and over multiple consecutive years.

For parlance, HERON tracks *resources* as commodities that flow between *components*. Components may be consumers, producers, or storage units; however, each component represents a single transaction (such as producing, consuming, or producing in order to consume). A single unit that has multiple transaction possibilities can be represented as multiple components.

The two primary use cases for HERON include system analysis under a regulated market system, and under a deregulated market system. In a regulated market system, a governing body takes ownership of all components and resources within a system and attempts to minimize system cost. In this manner, the operation of each component in the system is optimized for the benefit of the system in order to meet some required behavior such as disposing or providing a quantity of resource. In a deregulated market system, generally the requirement is not present, and the system is optimized for the most profitable configuration.

For example, power plants in a regulated system may be required to provide power to meet the demand of a local residential sector at the lowest possible cost. Power plants in a deregulated system choose when to provide power based on the prices offered in the energy market.

2.2 Capabilities

HERON provides two major functions: generation of RAVEN workflows for stochastic technoeconomic analysis; and dispatch models for optimal component operation given technical constraints and economic drivers.

2.2.1 Workflow Generation

HERON's primary utility is simplifying the creation of complex RAVEN workflows for grid-based stochastic technoeconomic analysis. In the past [1, 3] these workflows were created manually, often with significant difficulty for the user due to the interconnected and complex nature of the inputs. Instead, HERON provides a surrogate input via RAVEN's Templating system. HERON's input uses

terminology common to economic and energy system analysts, in contrast to RAVEN's input which uses terminology more common to statisticians and code analysts.

The fundamental RAVEN workflow is represented in Figure 1. There are two distinct layers in this workflow, an outer loop and an inner loop. The outer loop encapsulates changes to the sizing of components; that is, it manages changing the total capacity of components within the system. This management often takes one of two forms based on the analysis: *sweep* mode or *optimization* mode. In sweep mode, various combinations of component capacities are sampled from in order to obtain an understanding of how economic metrics change with respect to changes in the component capacities. This can yield differential or sensitivity results. In optimization mode, the outer loop seeks the cost-minimizing or profit-maximizing component capacities by exploring the variable space made up of component capacities.

The inner loop in Figure 1 seeks to determine the statistical behavior of economic metrics given a particular system configuration, i.e. fixed component capacities. Each inner loop begins with producing a set of stochastic histories. These histories may include market pricing, weather events, consumer demand, or any other uncertain phenomenon affecting the system of resources and components. For example, in analyzing a system of power plants regulated to meet demand, the demand from residential and commercial consumers may be represented as a stochastic process with seasonal behaviors. Similarly, when considering a system with wind power farms, the windspeed could be represented by a stochastic history. Once a realization of the stochastic elements is obtained, an optimization of component dispatch is performed to minimize cost or maximize profit of the system. This can be done with one of HERON's provided dispatching algorithms, or another can be supplied in its place. This is discussed further in Section 2.2.2. Finally, once an optimal dispatch is obtained for a given realization of the stochastic histories, a final economic determination is made for the economic metric of choice; for example, NPV, IRR, or PI. This single value for the economic metric may be considered one independent sample of the system selected in the outer loop. Many more samples are often needed for statistical convergence. Each sample begins with producing an independent realization of the synthetic histories, determining an optimal dispatch for those histories, then obtaining an economic determination for that realization.

The mathematical expression for the optimization process is given in [2] as follows:

$$M_{opt} = \min_{\vec{C},\vec{D}} \left(\mathbb{E}_{\omega} \left(M\left(\vec{C}, \vec{D}, \omega\right) \right) \right) \tag{1}$$

where:

- M is the economic metric of interest and M_{opt} is the optimal value of that metric;
- $\vec{C}$ is the set of capacities of the system components;
- $\vec{D}$ is the optimal dispatch of the capacities for a given realization ω;
- $\mathbb{E}_{\omega}$ is the expected value operator over realizations ω; and
- ω represents a single set of evaluations of the synthetic time histories.

Note that Equation 1 represents a cost minimization optimization; in the case of profit maximization, the minimization operator is trivially replaced by a maximization operator.

The algorithmic approach to this two-level problem is as follows:

1. Build a system of components and resources;
2. Select an initial configuration of component capacities;
3. For a predetermined number of desired realizations:
    a. Sample a set of synthetic histories;

b. Dispatch components to optimize economic metrics given selected component capacities and technical constraints;

c. Evaluate economic metrics for the optimal dispatch;

4. Collect statistics for economic metric evaluations;
5. Select a new configuration of component capacities based on statistical economic metric results;
6. Return to step 3 and repeat until desired convergence (optimal capacities or resolution of response space).

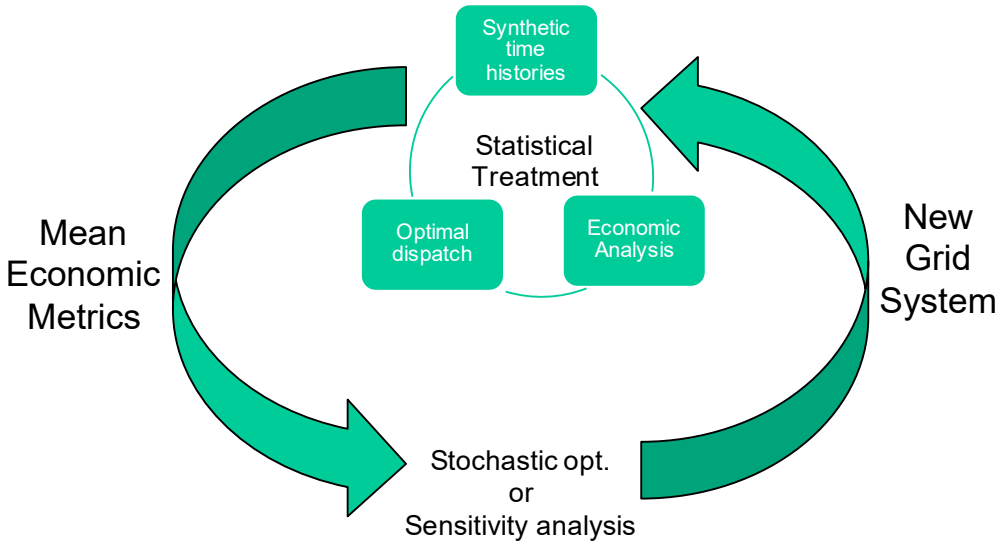

Figure 1: HERON's RAVEN-running-RAVEN workflow

An example of the format for the HERON input file is given in Figure 2. It structure is eXtensible Markup Language (XML). Under the root node there are three primary portions: “Case”, in which the global problem is established including the type of analysis to perform and general economics; “Components”, which define the technical and economic drivers for the system components; and “DataGenerators”, which define the sources from which external data will be obtained, such as Python functions and synthetic history generation (e.g. ARMA).

```
<EGRET>
  <Case name="Sweep_Runs">
    <mode>sweep</mode>
    <metric>NPV</metric>
    <num_arma_samples>2</num_arma_samples>
    <timestep_interval>3600</timestep_interval>
    <history_length>31536000</history_length>
    <economics>
      <ProjectTime>2</ProjectTime>
      <DiscountRate>0.08</DiscountRate>
      <tax>0.0</tax>
      <inflation>0.0</inflation>
      <verbosity>50</verbosity>
    </economics>
    <dispatch_increment resource="electricity">1e6</dispatch_increment>
  </Case>

  <Components>
    <Component name="BOP">
      <produces resource="electricity" dispatch="fixed">
        <capacity resource="electricity">
          <sweep_values>1095,1098</sweep_values>
        </capacity>
            <transfer>
           <Function method="power_conversion">myfirst</Function>
         </transfer>
      </produces>
      <economics>
        <lifetime>27</lifetime>
      </economics>
    </Component>

    <Component name="Electric_Grid">
      <demands resource="electricity" dispatch="independent">
        <capacity>
          <fixed_value>1e200</fixed_value>
        </capacity>
      </demands>
      <economics>
        <lifetime>3</lifetime>
        <CashFlow name="e_sales" type="repeating" taxable='True' inflation='none' mult_target='False'>
          <driver>
            <Function method="electric_source">exelon</Function>
          </driver>
          <reference_price>
            <fixed_value>0.1</fixed_value>
          </reference_price>
          <reference_driver>
            <fixed_value>1</fixed_value>
          </reference_driver>
          <scaling_factor_x>
            <fixed_value>1</fixed_value>
          </scaling_factor_x>
        </CashFlow>
      </economics>
    </Component>

  </Components>


  <DataGenerators>
  <ARMA name='Speed' variable="Signal">/ARMA/interpolated.pk</ARMA>
    <Function name="exelon">FCE_transfers.py</Function>
  <Function name="myfirst">FCE_transfers.py</Function>
  </DataGenerators>

</EGRET>
```

Figure 2: Example HERON Input

2.2.2 Technoeconomic Dispatch

Once a system configuration including resources and components is constructed, a particular set of component capacities is selected, and a realization of stochastic time series is obtained, HERON seeks to optimally dispatch the resources and components to meet system requirements and obtain the most desirable economic metric possible. There are several approaches to this dispatch optimization. ReEDS (Regional Energy Deployment System Model, developed at the National Renewable Energy Laboratory [14]) employs a time-slice approach using several representative days in a year to indicate dispatch trends as part of its capacity expansion model. While often sufficient to provide indicators for long-term capacity planning, this slicing approach may miss important technical limitations and behaviors observed in a higher-resolution time scale necessary for stochastic economic analysis; a higher-resolution consideration also allows improvements in storage management analysis that is not feasible in non-contiguous time slicing. EDGAR (Economic Dispatch Genetic AlgoRithms, developed at Argonne National Laboratory [13]) uses genetic algorithms to perform high-resolution dispatch optimization; however, its focus on electrical grid dispatch limits the application of the algorithms to arbitrary systems. However, these and other existing dispatch systems can be used in lieu of HERON's provided dispatching system with some code coupling.

The dispatching algorithms provided in HERON are intended to provide generic solvers for arbitrary systems at high resolution. The currently implemented generic dispatch algorithm leverages physics-based optimization and zero-finding to balance the system and obtain optimal component usage through frequent re-evaluation of cost drivers. This allows complex interactions of highly interdependent components to be optimized. This dispatch approach requires the user to provide transfer functions for components, describing the resource utilization that allows a component to consume or produce a resource. These transfer functions are written in Python [15] and allow extraordinary flexibility by providing access to the full system state at evaluation time. As such, these transfer functions can implement complex technical operation constraints. Because the high level of time resolution is maintained, the behavior of storage units is modeled on contiguous time, allowing increased fidelity in storage dispatching.

The genericity of HERON's nominal dispatcher comes at a cost, however. The continuous evaluation of the cost of particular system states takes notable computational effort, which results in a simulation that can be too slow to be practically used for many stochastic evaluations. To mitigate this, if a user knows the decision breakpoints for particular optimal system configurations, a Python-written dispatcher can be provided that solves optimal dispatch much more rapidly by limiting the number of evaluations and scenarios to be considered.

In the future, it is anticipated that additional dispatchers for HERON will be implemented to accommodate faster performance at the cost of complete genericity. For example, a dispatch optimization based on marginal dispatch, where the marginal cost of producing additional resources is assumed to be constant throughout the system evaluation, could potentially outperform the generic dispatcher significantly by drastically reducing the number of requisite cost evaluations.

An example of HERON dispatching results is provided in [16] and reproduced here in Figure 3. In this case, there are two resources tracked, electricity and hydrogen. There are 5 components: an electricity source (BOP), a hydrogen production facility (HTSE) that consumes electricity to make hydrogen, an electricity market (Electric_Grid) that consumes electricity, a hydrogen storage unit (H2_storage), and a constant hydrogen consumer (H2_market1). The graphs in Figure 3 show the operation of each unit in terms of electricity (top), hydrogen (middle), and storage as well as price of electricity (bottom). When electricity is sufficiently cheap, selling to the grid is not practical, so excess hydrogen is produced and stored. When electricity is sufficiently expensive, hydrogen is taken from storage to supply the hydrogen consumer so that all electricity produced can be sold at the grid. This performance in the three-day window shown suggests the general dispatch optimization results that can be expected from HERON's dispatching process.

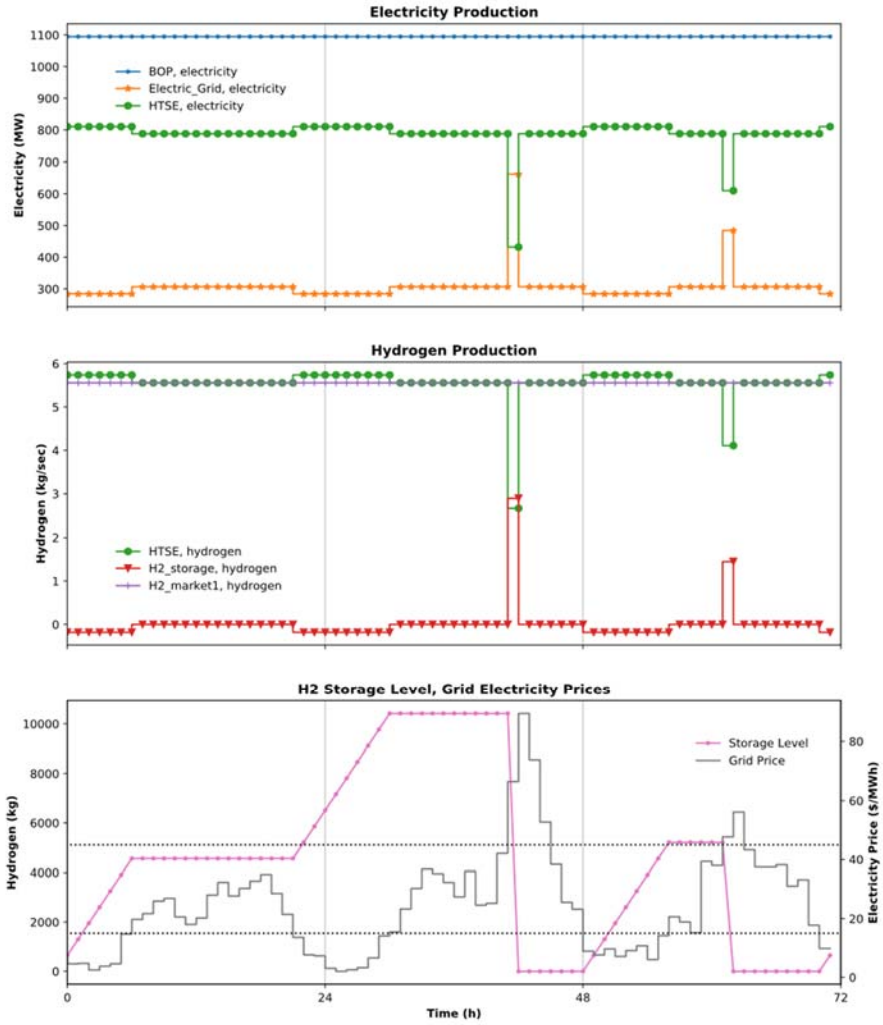

Figure 3: Example of HERON Dispatching results [16]

2.2.3 Example Dispatch

In interest of clarifying the process of HERON's generic dispatcher, we include an example with step-by-step consideration. In this example we consider the following five units:

- NPP: produces a constant stream of steam (600 units);
- BOP: consumes steam to produce electricity (consumes 10 units steam for 4 units electricity);
- HTSE: consumes electricity and steam to produce hydrogen (consumes 1 unit steam and 10 units electricity to produce 10 units hydrogen);
- HC: consumes hydrogen at a constant rate;
- Grid: flexibly consumes electricity.

The price of electricity sold at the Grid is given by a synthetic history [9, 10, 11] and changes for each time step. For this example, we consider on the resolution of the first time step as an example.

To solve this system, HERON employs a 4-step process for each hour in the simulation:

1. Dispatch fixed components completely.

2. Dispatch component minimum levels.
3. Balance resource utilization in the most economical way possible.
4. If desired, perturb independent sources to increase profit.

We follow this process through the example case. HERON begins by dispatching the fixed sources. This includes the NPP steam source and the HC hydrogen consumer. HERON updates its resource tracking table as follows, using positive as producing and negative as consuming for convention:

Unit	Steam	Electricity	Hydrogen
NPP	600		
BOP			
HTSE			
HC			-100
Grid			
Net	600		-100

Table 1: Fixed Component Dispatch

HERON then dispatches the minimum from each component. For instance, if a dependent or independent component has a ramp rate that limits its output at one hour based on the output at a previous hour, HERON dispatches the minimum given the ramp at this point. Additionally, if any independent or dependent component has a minimum production level, HERON dispatches it at this point as well. In this example case, we assume neither minimums or ramp rates influence the dispatch.

Once all fixed components are dispatched and the minimums from dependent and independent components are dispatched, HERON seeks to iteratively balance the resource utilization in the most economical way possible. It does this by seeking branching solutions to the existing imbalances. For example, in Table 1 there is an excess of steam and a deficit of hydrogen. HERON will consider ways to solve the imbalance; for instance, we start with the deficit of hydrogen. HERON inquires the components to see which can produce hydrogen. Since the HTSE is the only option, HERON dispatches the HTSE to fully provide the required hydrogen.

If at any point there is not a viable solution, HERON marks the existing scenario as not solvable, and considers alternative branching scenarios. If all scenarios fail, HERON will raise a Python exception (NotSolvableError) to indicate an impossible-to-balance scenario.

For the sake of this example, assume it takes 1 units of steam and 10 units of electricity to produce 10 units of hydrogen. To balance the deficit of hydrogen, 10 units of steam and 100 units of electricity are consumed in the HTSE. The current scenario updates as in Table 2.

Unit	Steam	Electricity	Hydrogen
NPP	600		

BOP			
HTSE	-10	-100	100
HC			-100
Grid			
Net	590		0

Table 2: Example Dispatch 1: Hydrogen Balance

Now the hydrogen has been balanced, but now a deficit of electricity has been introduced, and there is still an excess of steam. To balance the electricity deficit, the BOP consumes steam and produces electricity:

Unit	Steam	Electricity	Hydrogen
NPP	600		
BOP	-250	100	
HTSE	-10	-100	100
HC			-100
Grid			
Net	340	0	0

Table 3: Example Dispatch 2: Electricity Balance

With the electricity and hydrogen balanced, there remains only to balance the excess steam production. Following this same process, the BOP is requested to consume the excess steam to produce electricity, and the excess electricity is sold at the Grid, with a final result as shown in Table 4:

Unit	Steam	Electricity	Hydrogen
NPP	600		
BOP	-590	236	
HTSE	-10	-100	100
HC			-100
Grid		-136	
Net	0	0	0

Table 4: Example Dispatch 3: Full Balance

While this example is simple, it demonstrates the balancing behavior of HERON's generic dispatch optimization. If at any time

multiple components can consume or produce the same resource, HERON will consider the marginal cost of both scenarios. Notably, HERON does not trivially compare the marginal dispatch of only perturbing the components that directly support the balance effort, but also the downstream effects that are required to balance given choosing the component. For example, if two different components can produce hydrogen to meet a deficit, say one that consumes only electricity (HE) and one that consumes steam and electricity (HSE), HERON not only considers the marginal cost of either using the HE or HSE to produce hydrogen, but the cost of balancing the corresponding steam and/or electricity requirements (HSE). The comparison between scenarios is not made until a complete balance is made, to provide a fair comparison. In practice, branches tend to expand and collapse in straightforward manners.

For a branching example, consider Figure 4. In this example, a hydrogen storage has been added to the example case described above. Each green box represents a possible scenario, such as the tables above. Branching options from a scenario indicated multiple options to balance a resource. Only when the bottom level is reached (full balance) is a determination made between one option and another. For example, on the right branch, scenarios are generated until HERON can compare between the Market or Storage to consume Hydrogen. Red circles indicate that a scenario was accepted as the more economically viable, using marginal cost. Having selected Market, the algorithm moves back up one level and compares consuming electricity at the Grid or HTSE. Choosing the Grid, the algorithm then compares consuming steam at the BOP or the HTSE. All the choices downstream of the HTSE choice need to be made, then the two are compared, and in this example, the HTSE is chosen as the most economical option to balance steam.

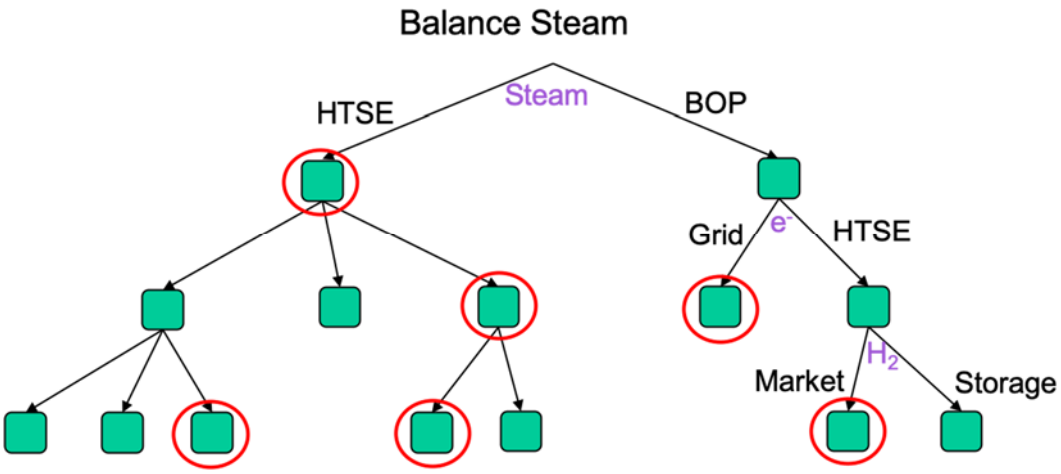

Figure 4: HERON Example Branching Scenarios

While pursuing this balance approach to dispatch optimization, the economic considerations and available performance of the components are constantly re-evaluated. While this introduces some computational cost, the benefit is an extraordinarily flexible evaluation process that allows complex interoperation between components in the system. Each scenario is tracked independently, so when transfer functions are inquired, the full state of the scenario in consideration is provided to the transfer function for consideration.

The genericity of this approach incurs a notable computational burden. Significant speed can be recouped by introducing assumptions. For example, assuming linear marginal costs and independence of the components would allow a one-time ordering of marginal costs for unit performance. We consider dispatch optimization algorithms with increased restrictions as

promising future work for HERON. Further, it is often not difficult for a user with Python experience to write a very efficient algorithm for dispatch that hinges on decision points that are only discovered implicitly by HERON's generic algorithm. This allows the fastest practical dispatch optimization and can greatly accelerate the technoeconomic analysis.

3. SOFTWARE INFRASTRUCTURE

3.1 RAVEN Plugin

HERON is not a stand-alone software; it is a plugin for RAVEN [8]. It requires RAVEN to run workflows and leverages several functionalities present in RAVEN. In this section the software and algorithms leveraged from RAVEN are discussed along with the structure of HERON itself.

3.1.1 Installation

As a plugin of RAVEN, HERON is installed as a submodule. RAVEN maintains up-to-date instructions for plugin installation in its manuals and other documentation. As of this writing, plugins are installed from the command line from within the top level of the RAVEN repository as

```
scripts/install_plugins.py -s heron
```

Note that RAVEN must be installed completely to use HERON and its components. At this writing HERON introduces one additional Python library to the standard set of RAVEN Python dependencies: `dill`. The serialization package `dill` enables information transfer between the HERON templating algorithm and dispatching algorithm not possible with the standard library set. This library, along with any future additional libraries required, should be installed correctly via RAVEN's documented installation procedure for including plugins.

HERON also requires use of the RAVEN plugin `CashFlow` [12], a utility for economic analysis. As of this writing, `CashFlow` is an officially supported plugin for RAVEN and is installed standard when plugins are included in RAVEN installation.

3.1.2 Plugin Additions

As a plugin of RAVEN, HERON currently provides External Models and Templates. The External Models are the dispatch algorithms included in HERON. The Templates are the tools needed to create the complex outer-inner RAVEN workflows to perform the technoeconomic analyses that are within HERON's scope. While RAVEN contains the base classes for both External Models and Templates, HERON maintains the specific implementations for its applications within its code repository.

3.1.3 HERON Structure

In accordance with RAVEN's requirements for plugins, the HERON repository consists of three main directories. The "src" folder contains the algorithms and software to perform its scope of work. The "doc" folder contains user manuals, design documents, quality assurance documentation, and any other documentation for the use, development, and maintenance of HERON. The "tests" folder contains regression tests to assure the acceptable performance of HERON to meet its software design description and associated use cases.

The software framework of HERON, contained in its "src" folder, contains an object-oriented framework as well as a driving executable. These are contained in Python modules that are described in the next several sections. Specific instructions for available options and their use are contained in the HERON user manual in the "doc" folder.

3.1.3.1 Cases

Cases describe the general simulations to be performed by HERON. It allows selecting options such as whether to perform *sweep* or *opt* calculations on the outer loop, the economic metrics of interest, the number of inner samples to take per outer iteration, and so forth. It also allows specification of economic factors such as the length of time to consider in evaluation as well as discount rates, tax rates, and inflation rates. In this way, a Case contains the general information about the simulation.

3.1.3.2 Components

Components describe the passage and utilization of resources within the HERON simulation. Each component is comprised of two parts: a technical definition, and an economic definition. The technical

definition depends on the activity of a unit. For example, a Component's activity can be "produces" if it either consumes one resource to produce another or simply yields a resource, "demands" if it only consumes a resource, or "stores" if it is meant to act as a storage for a particular resource.

Each Component also indicates its place within the dispatch optimization. A Component with "independent" dispatch may be freely perturbed by HERON when it seeks to take profitable action. For example, if there is excess steam during computation, HERON will consider all Components with "independent" dispatch who consume steam as viable solutions to absorb the excess. Components with "dependent" dispatch cannot directly be controlled by HERON but may operate flexibly to respond to changes in other "independent" Components. For example, if increasing steam consumption in an "independent" Component results in excess electricity, a "dependent" Component who consumes electricity may be considered as a candidate to absorb it. Finally, "fixed" dispatch Components always perform exactly at their capacity. A "fixed" Component that "demands" always demands a consistent quantity of resource and may not be adjusted by HERON to balance resource usage, while a "fixed" component that "produces" always provides a consistent quantity of resource and may not be adjusted by HERON.

In additional nodes within the activity description can specify the capacity of the Component, the transfer function for unit operation, minimum production levels, and such. The units for all Component descriptions is assumed to be a rate; that is, a Component who "produces" steam with a capacity of 100 is assumed to produces 100 units of steam per time interval. The specific units are not specified; the burden of consistent unit usage is on the user. However, if a Component "stores" a resource, it assumes units of capacity are given in quantity, not rate. For example, a Component that "stores" steam with a capacity of 100 is assumed to be able to hold up to 100 units of steam at one time (not 100 units per time interval). Note that each value-based entry (capacity, minimum, etc) is specified through a ValuedParam option (see Section 3.1.3.6).

After specifying the activity of a component, an "economics" node is used to declare the economic interactions of the component. These are specified by defining "CashFlow" nodes, corollaries to those found in the `CashFlow` [12] plugin of RAVEN. Each CashFlow node is specified as a "one-time" type or "repeating" type, corresponding to capital investments and regular transactions respectively. Each of the elements of the CashFlow (driver, reference price, reference driver, and scaling factor) is specified as a ValuedParam (see Section 3.1.3.6).

3.1.3.3 DispatchScenario

Dispatch Scenarios are tools internal to HERON for considering diverging options in dispatch optimization. When performing dispatch and finding multiple suitable options for increasing profitability or balancing resource, HERON will create a Dispatch Scenario for each option, inheriting the initial state from the Scenario before it and choosing one action or the other. This allows a broad tree of actions to be considered, allowing HERON dispatchers to select the most optimal scenario given complete resolution of the possibilities. The process for spawning and collapsing these scenarios is discussed further in Section 2.2.2. Dispatch Scenarios are used exclusively within HERON and are not directly accessible to the user via API.

3.1.3.4 Placeholders

Many of the activities of Components in a simulation may be governed by generated synthetic histories or functional evaluations that are not possible to evaluate when writing the HERON input but become available at run time. To accommodate these input placeholders, a class of Placeholders was implemented with specific definitions for both synthetic history (ARMA) and Python function (Function) objects. In the HERON input file, a section is reserved for "DataGenerators" in which RAVEN synthetic history generators and Python function modules can be indicated for use within the simulation. These are then used within the ValuedParams system (see Section 3.1.3.6) to provide placeholder values evaluated at run time.

3.1.3.5 Economics

To encapsulate the economic attributes of Components separate from the activities they perform, the Economics class exists to provide tools to track and interact with the CashFlow plugin. These read in the "economics" parts of the HERON input in both Cases and Components. They also hold the algorithms to write the full CashFlow inputs once a particular Dispatch Scenario is selected for a time interval.

3.1.3.6 ValuedParams

Because HERON's generic dispatcher intends to provide an extraordinarily flexible algorithm, ValuedParams were developed to provide a generic API for input values that could come from a variety of sources. A ValuedParam input entry has one of the following forms:

- fixed_value, for when a value is a constant float value;
- sweep_values, for discrete values to iterate over in *sweep* mode;
- opt_bounds, for bounding limits in *opt* mode;
- ARMA, for evaluating an entry in a synthetic history;
- Function, for obtaining a value from a function;
- variable, for taking a value from RAVEN's variables.

"fixed_value" takes a single number entry, while both "sweep_values" and "opt_bounds" take a comma-separated list. "variable" takes the name of a RAVEN variable, while "ARMA" and "Function" take a description of the synthetic history generator or Python function along with the variable name to retrieve.

3.1.3.7 dispatch

The "dispatch" module contains the algorithms to carry out generic dispatch under economic drivers and technical constraints, leveraging the other classes in HERON. It exercises its physics-based balancing dispatch optimization on successive time windows and steps, selecting the preferable dispatch scenarios until the simulation is complete. This dispatcher is designed to be compatible with interpolated, clustered ARMA surrogates trained by RAVEN, solving representative segments and applying weights to economic factors to approximate full-year simulations. This dispatch algorithm dynamically constructs objects from the CashFlow plugin and runs economic evaluations.

3.1.3.8 input_loader

The input loader is responsible for reading the HERON XML-based input and providing summary information about the contents.

3.1.3.9 main

The driver for HERON, main is responsible for engaging the input loader, instantiating HERON objects based on the user input, and engaging the RAVEN Template system to construct RAVEN workflows.

3.1.3.10 templates

Inheriting from the RAVEN base Template classes, the structures in the "templates" folder load existing templates, populate them with the correct names and objects based on the HERON system loaded by the driver, and write completed workflow files that can be run by RAVEN. More information about templating can be found in the RAVEN documentation [8].

3.2 Version Control and Accessibility

HERON is currently maintained as a Git repository hosted on INL's internal GitLab [17] repository. While not publicly accessible, access can be obtained by applying for and receiving an account with the INL High Performance Computing (HPC) group. Upon acquiring credentials, INL's GitLab repositories can be accessed locally on INL campus or remotely.

As described in the SQA documentation (see Section 4.1), development work is done on development branches and must undergo review, testing, and approval before being pushed to the master branch of HERON. This enforces control of issue resolution and feature development.

4. QUALITY ASSURANCE PRACTICES

4.1 SQA Documentation

As a plugin of RAVEN, HERON inherits its Software Quality Assurance (SQA) practices and documentation from RAVEN. Any deviations from RAVEN's base SQA are documented in HERON's SQA documentation. The documents outlining HERON's specific SQA are:

- Configuration Items List: The CI List aims to gather Software, Hardware, and Documentation components of the Project's infrastructure. These are the items that define the configuration of the software (see appendix in Section 9).

- Software Design Description: SDD is a document written by Software Designer/Architect to give other Software Developers an overall guide to the architecture of the software product. It has an architecture diagram with detailed information on System Purpose and Scope, along with the Structure of Code and Software. It serves as a reference document outlining all parts and functionality of the software (see appendix in Section 8).

- Software Requirement Specification: SRS has a complete description of the features and behavior of the system. The document includes various elements describing Functional, Usability, Performance, and Minimum requirements of the software. It also captures how the system interfaces and operates with external applications (see appendix in Section 8).

- Requirement Traceability Matrix: RTM is a document that co-relates user requirements with test cases to validate the completeness of the software. The parameters included in RTM is the Requirement ID, the Requirement Description, and the Test Case for verification of every requirement (see appendix in Section 8).

4.2 Regression Testing

HERON inherits the same regression testing philosophy and framework used by RAVEN; that is, before any modifications to the code can be merged into the main development branch, a series of regression tests must be passed to assure behavior has not notably changed; or if it has changed, the changed behavior is documented and updated as part of the code changes. These regression tests are a combination of unit tests, which test the performance of individual sub-algorithms within HERON, and integration tests, which test the ability of HERON to perform as a whole to satisfy particular use cases. At the time of this writing there are four regressions tests in HERON. These test configurations are all available in the code repository under `heron/tests`.

To demonstrate the capability of the software simple and analytically tractable test cases were needed. Three such cases were designed with a view to demonstrate the capabilities of the software rather than the practical applicability of the numbers obtained. With this view in mind the following three scenarios were considered.

4.2.1 Variable Demand with Fixed Pricing

A simple test case was built with a source feeding into a power producer which in turn is feeding into an electrical grid which has the ability to absorb all the power produced by the producer. For this test case to work the components in Figure 5 were put together into the Heron input files.

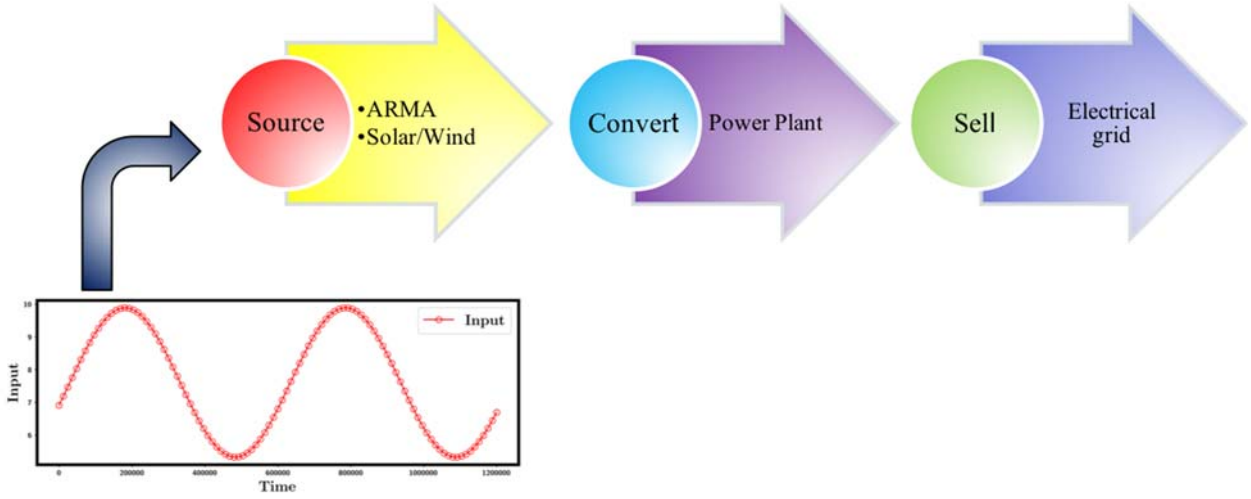

Figure 5: Regression Test 1 System

a) Source: A sinusoidal wave was generated by training an ARMA on a fictitiously generated sine wave. The result of the training is a perfectly generated deterministic sinusoid.
b) Power Plant: This component was modeled by a transfer function module which essentially takes the synthetic signal and converts it into a power signal.
c) Electrical grid: This component is a market and is essentially an infinite sink for electricity produced.

The Net present value (NPV) was selected as a metric. Which is calculated at 20 sampled points. It was calculated both analytically and through the Heron software package for a period of 2 years. For this cashflow at each point was calculated as

$$CF = Power * selling\ price\ (in\ \$).$$

NPV was then computed as:

$$NPV = \sum_{y} \frac{CF_t}{(1+d)^y}\ , y = years, d = discount\ rate$$

where y is the year of project analysis and d is the discount rate. The selling price of electricity used was \$0.1/MW, and the discount rate was 0.08. The analytic result for this configuration is an NPV of \$35.66, which HERON obtains exactly.

4.2.2 Variable Demand with Variable Pricing

A slight tweaking of the first case was attempted where the new market doesn't have a fixed price, as in Figure 6.

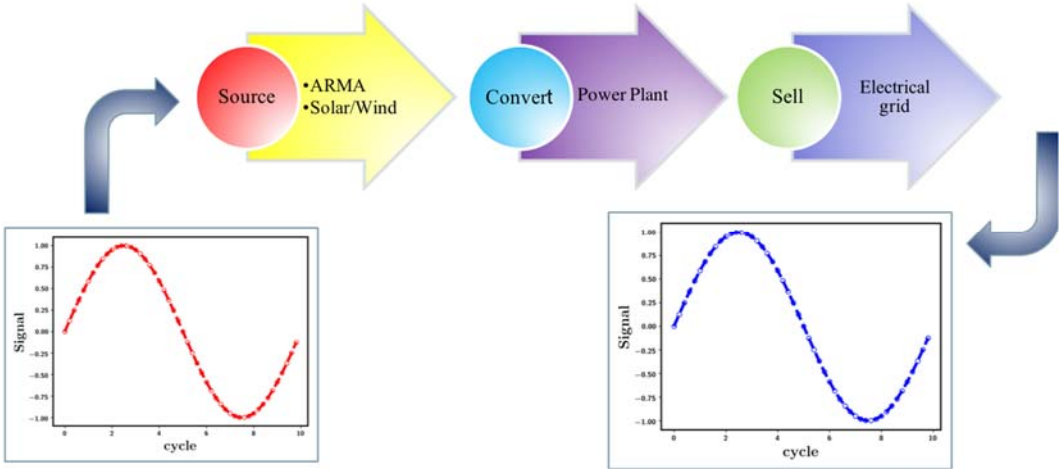

Figure 6: Regression Test 2 System

The mechanics of this test case is very similar to the first one with a difference that the price of the electricity in the grid is varying through a sinusoidal curve. The NPV metric was again computed both analytically and through HERON. The analytic value for the NPV in this configuration is $17.83, which HERON obtains exactly.

4.2.3 Multiple Markets with Fixed Pricing

A more generic case was designed (Figure 7) with two electrical markets to choose from, in which the capacity of the better paying market is limited. For this current case the capacity of the better paying market is set to 1 while the price of electricity is set to $ 10. The available price of electricity in this additional source of consumption is significantly higher than the default market where the available price is still $ 0.1. With this the NPV metric was again computed. The analytic value is $388.75, which HERON computes exactly.

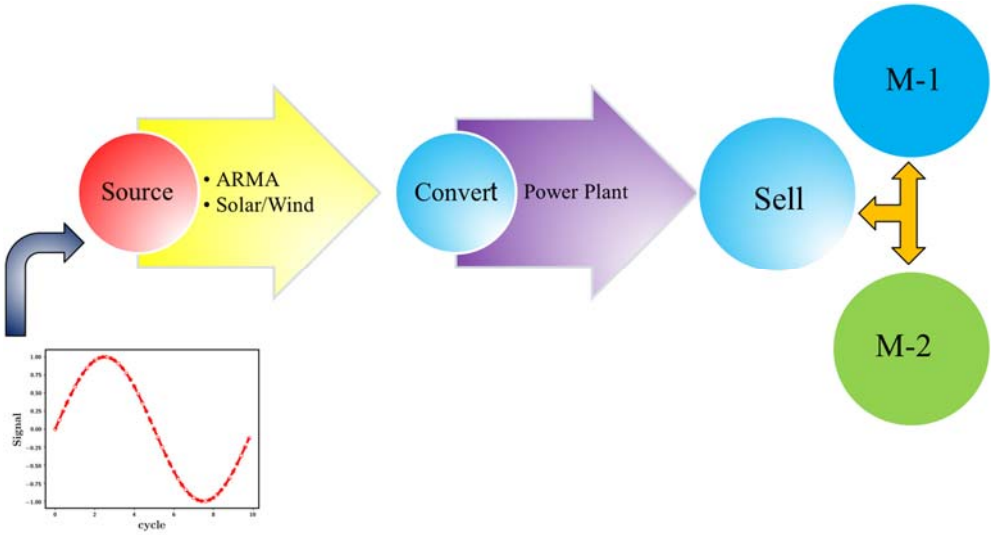

Figure 7: Regression Test 3 System

5. DEMONSTRATION CASE

5.1 Motivation

Technoeconomic analysis of grid energy systems may be classified into two systems: those with regulated markets, and those with deregulated markets. As mentioned, in a regulated market system, a governing body takes ownership of all components and resources within a system and attempts to minimize system cost. In this manner, the operation of each component in the system is optimized for the benefit of the system in order to meet some required behavior such as disposing or providing a quantity of resource. In a deregulated market system, generally the requirement is not present, and the system is optimized for the most profitable configuration.

For example, power plants in a regulated system may be required to provide power to meet the demand of a local residential sector at the lowest possible cost. Power plants in a deregulated system choose when to provide power based on the prices offered in the energy market. Since deregulated markets make up a notable portion of energy markets in the U.S., we demonstrate HERON to such a case as evidence of its efficacy in these kinds of analyses.

5.2 Case Definition

To demonstrate HERON's application to deregulated markets, we document a simple case involving five components:

- A generic power plant (EPP) that produces electricity at a cost;
- A hydrogen production plant (HPP) that consumes electricity to produce hydrogen;
- A hydrogen storage unit (HS) that may receive or provide hydrogen;
- An electric grid (EG) as a selling market for electricity; and
- A hydrogen selling market (HM).

For this case, we hold the EPP at a constant production level. We sweep over the capacity of the HPP, HS, and HM to understand the economic impacts of changing these parameters. We allow the EG to be functionally an infinite sink for electricity production. See Table 5 for the specific values used in the analysis.

Component	Consumes	Produces	Rate	Capacity
EPP	-	MW	infinite	1000 MW
HPP	MW	H_2	100 MW per 1 kg/s H_2	1, 5, 10 kg/s H_2
HS	H_2	H_2	infinite	3.6, 18, 36 1000kg H_2
EG	MW	-	Infinite	infinite
HM	H_2	-	infinite	1, 5, 10 kg/s H_2

Table 5: Deregulated Case Definition

The market price for hydrogen is derived from supply/demand curves in [37], while the price for electricity is taken stochastically from a RAVEN synthetic history ROM trained on the PJM ISO 2018 electricity price data [38]. Other economic factors are taken from data and figures in [37]. For operating the hydrogen storage unit, we set the buy threshold to \$25/MWh and the sell threshold to \$50/MWh, corresponding to the electricity prices which trigger the storage to acquire or release hydrogen respectively.

5.3 Synthetic History Training

The Pennsylvania-New Jersey-Maryland Interconnection (PJM), an ISO in the U.S., operates as a deregulated market. This makes data from this interconnect suitable for training example synthetic economic data for energy prices. In this demonstration we use the 2018 PJM electricity price data [38], as shown in Figure 8. Of interesting note, a "bomb cyclone" affected the PJM region in 2018, bringing very cold temperatures and driving up the demand for electricity.

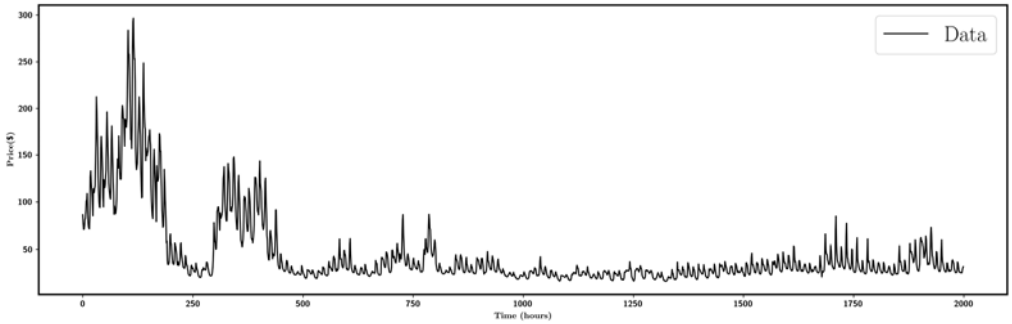

Figure 8: PJM 2018 Electricity Prices

According to RAVEN's requirements for synthetic history training, the electricity price was trained with Fourier modes corresponding to yearly, semiannual, 3-month, day, half-day, 4 hours, and 3 hours Fourier modes; ARMA (P, Q) values of (2, 1), and segmenting daily with 45 clusters. While we represent a 20-year project life, we do not vary the mean or distribution of the electricity price from one year to another. A set of samples imposed on the original training data is given in Figure 9. While there is room for improvement in the training of this algorithm, it is sufficient for this demonstration case. The comparison between the cumulative distribution functions of the sampled histories and the original sample is given in Figure 10.

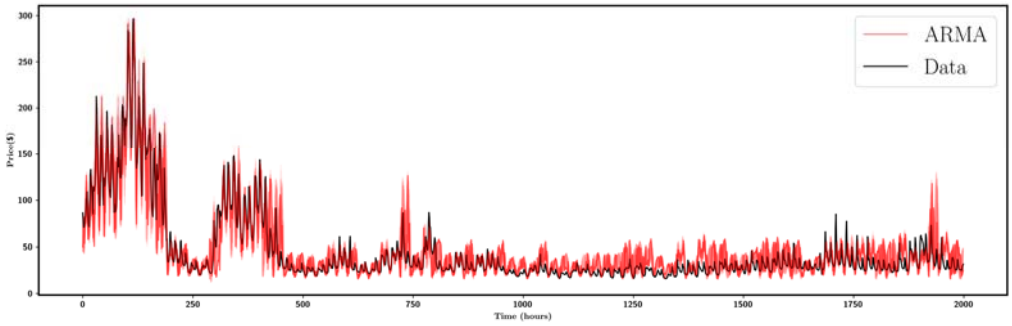

Figure 9: PJM Synthetic Signals Sampled

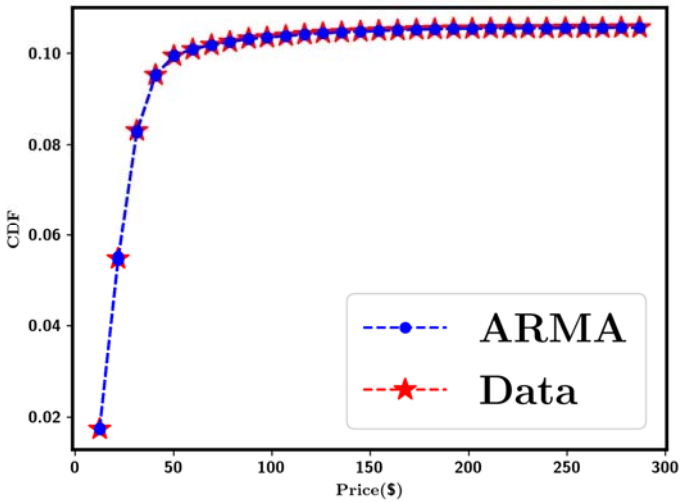

Figure 10: CDF comparison for synthetic data

5.4 Case 0: Nominal Case

In the nominal case, HERON performed the sweeping technoeconomic analysis as described above. The optimal dispatch discovered by HERON is shown in Figure 11for 10 days starting the second week of January for the case with a HPP capacity of 10 kg/s, a HS capacity of one million kg, and a HM capacity of 5 kg/s. In these three plots, the top shows the dispatch of electricity, the middle shows the dispatch of hydrogen, and the bottom combines the electricity price and hydrogen storage level. As expected, there are three types of behavior, depending on the price of electricity:

- When the price of electricity is high, all available electricity is sold entirely to the EG, unless insufficient stored hydrogen exists to supply the HM; in that situation, only the minimum electricity amount to satisfy the user is sent to the HPP to satisfy the HM.
- When the price of electricity is low, all electricity possible is diverted to production and storage of hydrogen in the HS.
- When the price of electricity is between the battery buy-in and sell-out prices, the electricity is used at the HPP just sufficiently to satisfy the HM, and the remainder is sold at the EM.

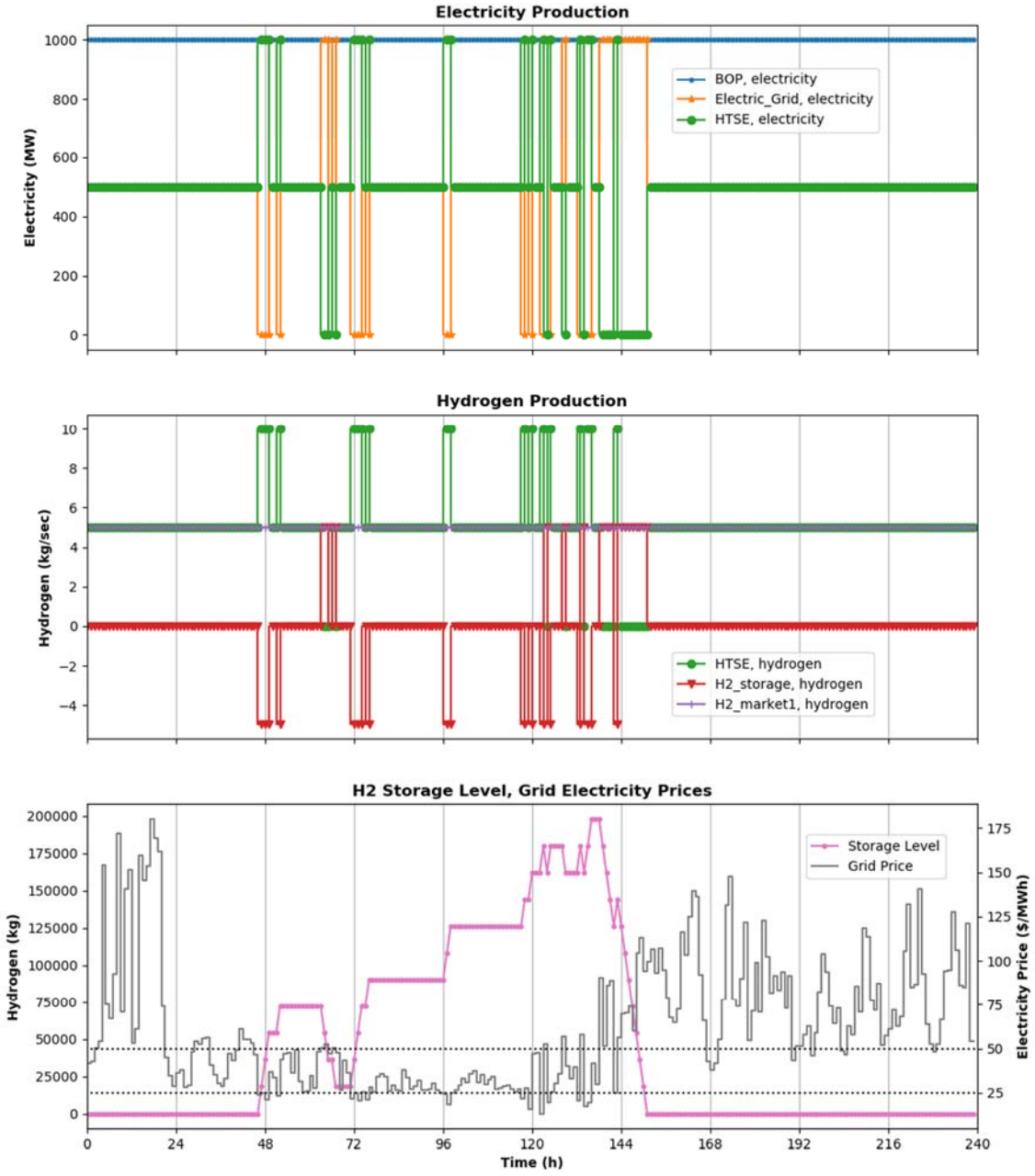

Figure 11: Example Dispatch Optimization

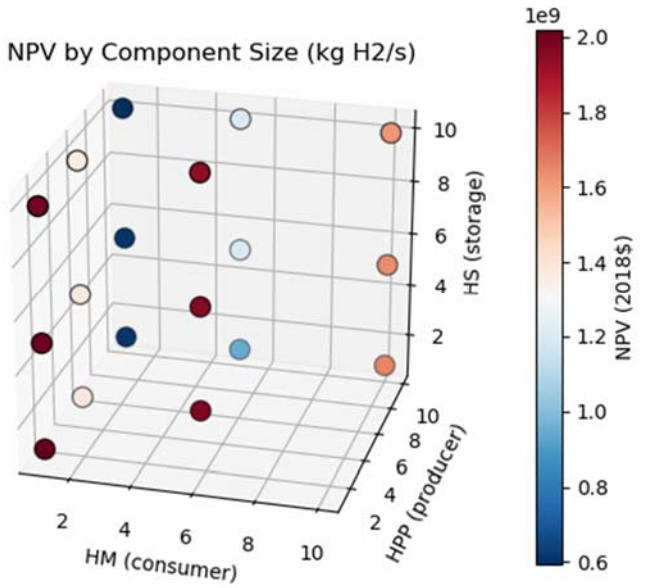

Figure 12: Case 0 Sweep Results

Table 6: Case 0 Sweep Results

HM size (kg/s)	HPP size (kg/s)	HS size (1000 kg)	mean NPV (B 2018$)
1	1	3.6	2.02
1	1	18	2.01
1	1	36	1.99
5	5	3.6	1.97
5	5	18	1.95
5	5	36	1.94
10	10	3.6	1.65
10	10	18	1.64
10	10	36	1.62
1	5	3.6	1.38
1	5	18	1.38
1	5	36	1.36
5	10	18	1.20
5	10	36	1.20
5	10	3.6	0.94
1	10	3.6	0.61
1	10	18	0.61
1	10	36	0.59

The full results of the stochastic sweep are shown in Figure 12 and Table 6 (sorted by NPV). The maximum mean NPV is discovered when the HPP, HM, and HS are all small. NPV drops slowly as the size of the storage increases and drops quickly as the size of the HPP increases. Some of this loss is recouped as the size of the HM is proportionately increased. The motivation for minimal hydrogen involvement may be driven by a low profitability of the hydrogen. Additionally, the storage is not sufficiently enabling the electricity from the EPP to be sold during peaks in the electricity price at the EG. This may be improved with more volatile electricity pricing. Given these two considerations, we introduce changes in the following cases to the hydrogen price and electricity price signal to consider the sensitivity of HERON to these perturbations.

5.5 Case 1: Increased Hydrogen Price

For this case we consider increasing the value of hydrogen across the board by 50%, leaving all other parameters the same. The results are show in Figure 13 and Table 7. Due to the increase in the value of hydrogen, the overall NPV is much larger. Additionally, the optimal configuration is to maximize the production and market size for hydrogen. However, the storage is still not sufficiently motivated; the optimal configuration minimizes storage size.

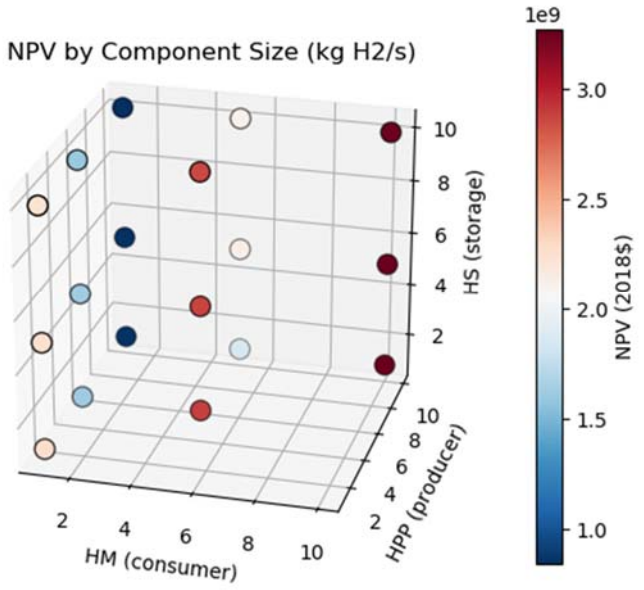

Figure 13: Case 1 Sweep Results

Table 7: Case 1 Sweep Results

HM size (kg/s)	HPP size (kg/s)	HS size (1000 kg)	mean NPV (B 2018$)
10	10	3.6	3.27
10	10	18	3.26
10	10	36	3.25
5	5	3.6	2.89
5	5	18	2.88
5	5	36	2.86
1	1	3.6	2.27
1	1	18	2.26
1	1	36	2.24
5	10	18	2.13
5	10	36	2.12
5	10	3.6	1.87
1	5	3.6	1.63
1	5	18	1.62
1	5	36	1.61
1	10	3.6	0.86
1	10	18	0.85
1	10	36	0.84

5.6 Case 2: Increased Electricity Price Volatility

The hydrogen storage (HS) in the proposed system is what enables a deregulated behavior. Since the hydrogen market (HM) demands a constant stream of hydrogen, the system can only flex to sell additional electricity to the electricity grid (EG) by taking hydrogen from the HS to satisfy the HM. This will be most beneficial when there are frequently oscillating periods of low electricity prices, during which excess hydrogen is produced and stored, and high electricity prices, during which all electricity is sold at the EG. Also, the hydrogen production plant (HPP) must be sufficiently larger than the HM so that hydrogen can be stored and enable rerouting electricity from the HPP to the EG during peak prices. Finally, the HS must be large enough to enable electricity price peak finding.

Case 0 and Case 1 showed no improvement (in fact, slight deterioration) of the NPV as a result of increasing HS size. The potential benefits of HS sizing appear small compared to the effects of sizing HPP and HM. To illustrate the sensitivity of the NPV to HS sizing, we select a particular size of HM and HPP and survey various HS sizes and their resulting NPV. We fix the HPP size at 10 kg/s H2 and the HM at 5 kg/s. Several samples of the increased-volatility signals are shown in Figure 15. Note that the prices both peak higher and drop lower than in the original signal (Figure 9).

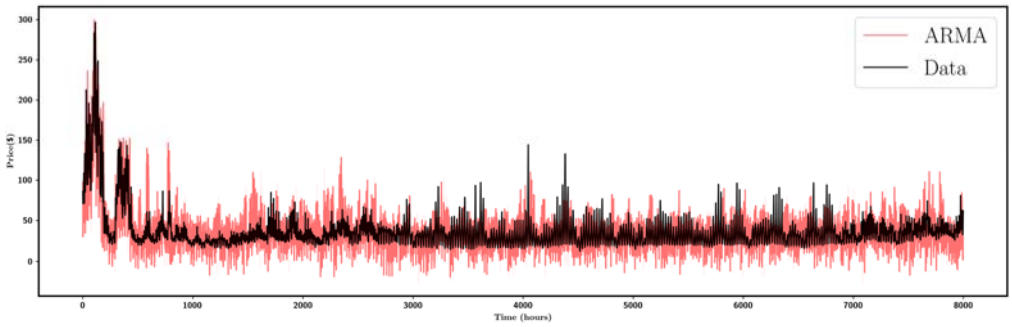

Figure 15: High-volatility signal

Table 8 and Figure 14 show the results of several different storage sizes and their impact on the NPV. Additional sampling has been taken near the profitability turning point.

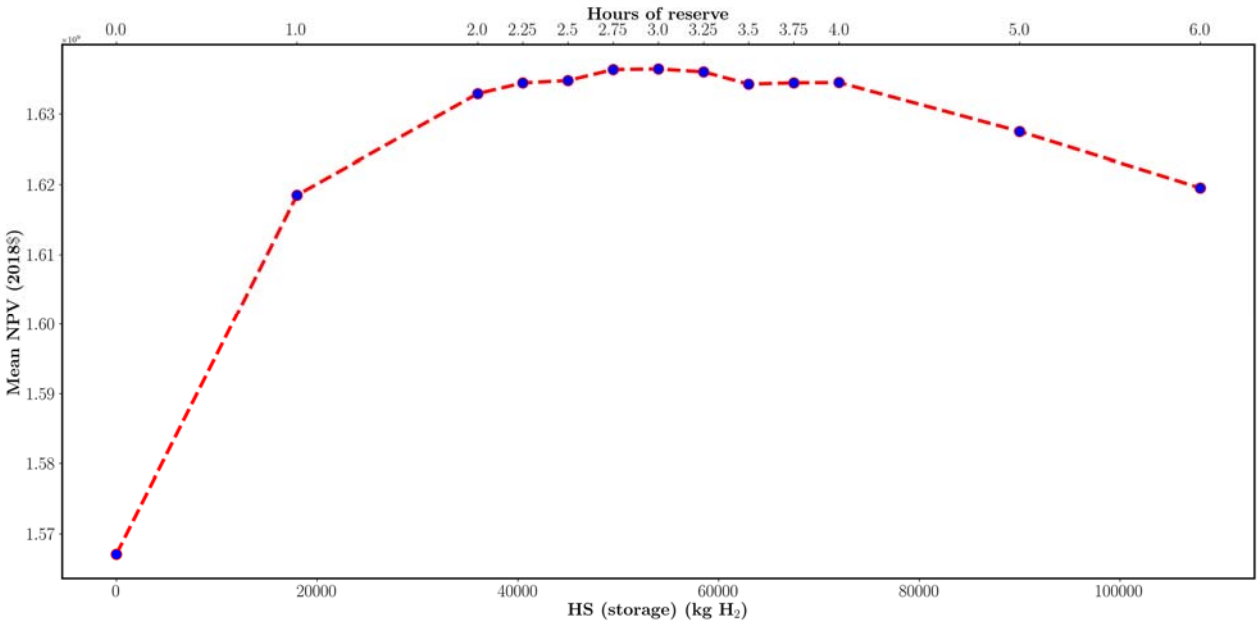

Figure 14: Case 2 Sweep Results

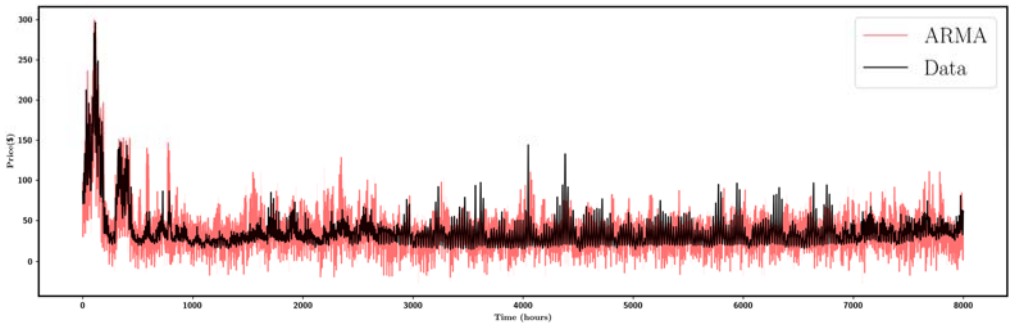

Figure 15: High-volatility signal

Table 8: Case 2 Sweep Results

HS size (hours reserve)	HS size (1000 kg)	mean NPV (B 2018$)
0	0	1.567
1	18000	1.618
2	36000	1.633
2.25	40500	1.634
2.50	45000	1.635
2.75	49500	1.636
3	54000	1.636
3.25	58500	1.636
3.50	63000	1.634
3.75	67500	1.634
4	72000	1.634
5	90000	1.628
6	108000	1.620

The results indicate there is an expected optimal HS size for a given HPP and HM size. For an HPP with capacity 10 kg/s and HM with demand of 5 kg/s and given the highly volatile price signal given as well as the storage buy-sell prices used (25 and 50 $/MW respectively), this optimum appears to be around 50,000 kg, or 3 hours' worth of HM demand; that is, the HS at full capacity can deliver the needs of the HM for nearly 3 hours. We expect this to change critically with the HPP and HM size, the price signal, and the buy/sell storage threshold prices.

5.7 Case 3: Optimization

Having explored the impacts of volatility and hydrogen pricing on HERON's performance, we consider changing from a *sweep* mode, in which the goal is to understand a response surface, so an *optimization* mode, where the goal is to find the maximum mean NPV across the component sizing domain. HERON leverages RAVEN's optimization algorithms for this; in this case we use RAVEN's SPSA optimization algorithm. We return to the Case 0 conditions for this study. The bounds of the optimization search are given in Table 9, and the target is maximizing the mean NPV. We constrain the optimization to only consider points in which the HPP size is at least the HM size.

Table 9: Case 3 Optimization Bounds

Feature	HPP size (kg/s)	HM size (kg/s)	HS size (1000 kg)
Lower bound	0	0	0
Upper bound	2.5	2	27
Initial Point	0.2	0.1	1
Optimal Point	0.33981824	0.31610014	0.999979054

Figure 16 shows the optimization path taken by RAVEN, after the problem was set up using HERON. The red point is the initial sample point; accepted points are represented as line-connected dots colored by the mean NPV; and rejected optimization points are shown as crosses. As can be seen, the optimization improved the mean NPV by roughly one hundred thousand 2018$ by searching the domain. The optimal sizing was discovered by increasing the HPP size by 0.14 kg/s, increasing the HM size by 0.216 kg/s, and maintaining the HS size in 85 iterative steps. The optimal point suggests keeping roughly an hour of hydrogen in the HS storage and having an HPP slightly bigger than the HM.

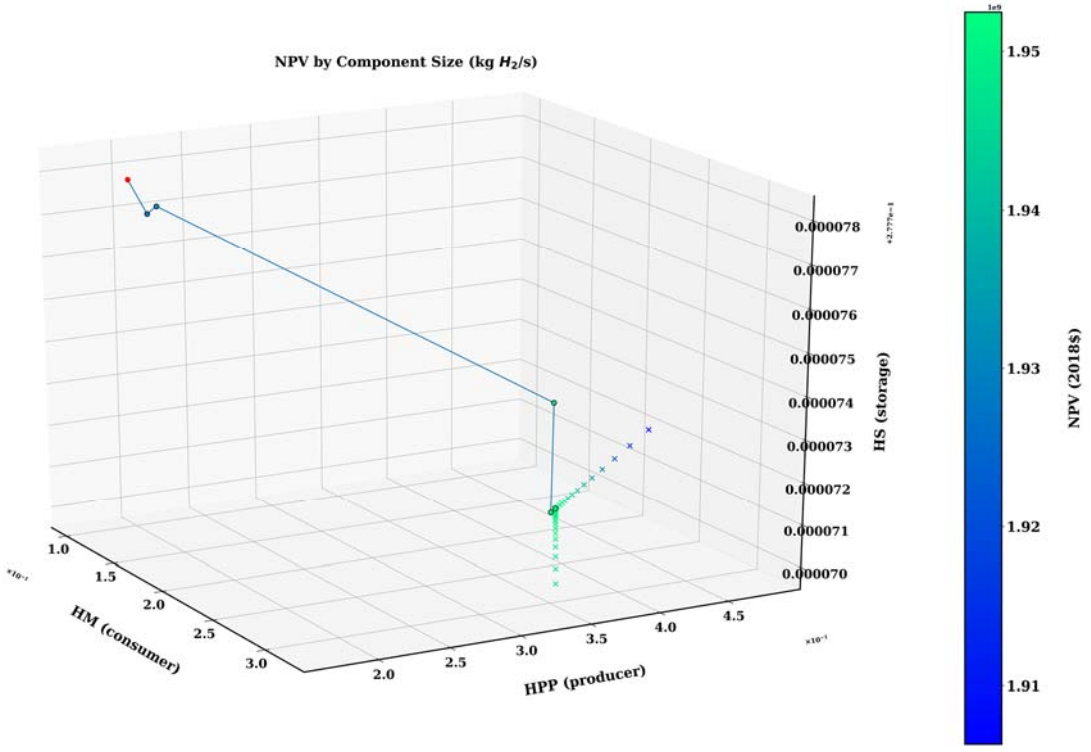

Figure 16: Case 3 Optimization Path

6. CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

This document reports the development of HERON for application to deregulated markets. The motivation for continuing this novel approach to stochastic technoeconomic analysis has been discussed, and approaches from similar toolsets have been considered. The software design for HERON has been presented, including the intended use as a RAVEN workflow generator and dispatch optimization engine. The process for performing generic technoeconomic dispatch has been outlined, and potential improvements suggested. Integrating the technoeconomic dispatch within stochastic optimization as performed by RAVEN has also been discussed.

The internal software design of HERON has also been presented, including its relationship as a plugin of RAVEN, the new features added including Templates and External Models, and the new workflows enabled via HERON. The design philosophy and data flow in HERON has been presented, and the means to access the plugin have been outlined. Additionally, the quality assurance plan for HERON has been discussed, including software quality assurance practices and documentation. The initial set of HERON regression tests for continuous integration assurance have been outlined, and their merit as demonstrate of use cases explained.

Finally, a series of demonstration cases applying to deregulated market analysis have been examined. Starting with a nominal case connecting a power supplier, a hydrogen producer, hydrogen storage, an electricity market, and a hydrogen consumer, the effects of changing a variety of parameters on the behavior of HERON have been considered. Through these demonstration cases, we build confidence in the use of HERON as a toolset for technoeconomic analysis of grid-energy systems in deregulated markets.

6.2 Ongoing Work

There are many avenues of exploration still available to the ongoing improvement of stochastic technoeconomic analyses with HERON and RAVEN. Improvements consist of efficiency increases in the code as well as expanded regression test coverage and development to address additional test cases.

As noted, the existing generic dispatch is powerful in its ability to handle complex interactions but limited by its slow operation. Some promising approaches include adding a linear margin-based dispatcher, which assumes a constant marginal price for components and evaluates them once instead of at every consideration. This could greatly accelerate the dispatch optimization process. Further, a decision-point-based generic dispatcher could allow the user to program particular behaviors based on system evaluation, which could also greatly accelerate the dispatch while maintaining a reasonable level of genericity.

Further development of HERON is desirable to expand coverage into multi-level markets. For instance, in many U.S. energy markets the dispatch of electricity is divided into day-ahead hourly markets and hour-ahead 5-minute markup markets. The day-ahead hourly markets indicate expected behavior and commitments from units significantly before the dispatch, while the hour-ahead 5-minute markets are a near-real-time response to actual load behavior. Often, a component producing electricity may desire to reserve some capacity to contribute on the 5-minute market, increasing its economic viability. This multi-level market feature would increase the accuracy and flexibility of HERON to consider such systems.

A subject of recent interest is ancillary services and markets provided by electricity producers. These ancillary markets, such as capacity, spinning reserve, and frequency stability, could potentially highlight the value of some components that is not captured in a simple market strategy. Similarly, economic policy can often greatly influence the most economical system configuration; a simple way to include such policies in HERON would likewise expand its effectiveness for such systems.

One of the assumptions of the generic dispatch optimization algorithm in HERON for regulated cases is that the load demand must be met. Often, this is not strictly true in terms of governance; rather,

the load demand must be met to a particular degree of satisfaction and given a certain tolerance for risk. Extending HERON to consider robust optimization as a method to address risk-weighted technoeconomic dispatch would allow it to analyze such system configurations.

7. REFERENCES

1. C. Rabiti, A. S. Epiney, P. W. Talbot, J. S. Kim, S. Bragg-Sitton, A. Alfonsi, A. Yigitoglu, S. Greenwood, S. M. Cetiner, F. Ganda, G. Maronati, "Status Report on Modeling and Simulation Capabilities for Nuclear-Renewable Hybrid Energy Systems," Idaho National Laboratory, September 2017, INL/EXT-17-43441.
2. A. Epiney, C. Rabiti, P. Talbot, A. Alfonsi, "Economic analysis of a nuclear hybrid energy system in a stochastic environment including wind turbines in an electricity grid", Applied Energy, accepted 28 November 2019.
3. A. Epiney, C. Rabiti, P. W. Talbot, J. S. Kim, J. D. Richards, S. Bragg-Sitton, "Case Study: Nuclear-Renewable-Water Integration in Arizona", Idaho National Laboratory, September 2018, INL/EXT-18-51359.
4. A. S. Epiney, J. Chen, C. Rabiti, "Status on the Development of a Modeling and Simulation Framework for the Economic Assessment of Nuclear Hybrid Energy Systems (FY 16)", Idaho National Laboratory, September 2016, INL/EXT-16-39832.
5. A. S. Epiney, R. A. Kinoshita, J. S. Kim, C. Rabiti, M. S. Greenwood, "Software development infrastructure for the HYBRID modeling and simulation project", INL report INL/EXT-16-40004, September 2016.
6. J. S. Kim, H. E. Garcia "Nuclear-Renewable Hybrid Energy System for Reverse Osmosis Desalination Process," Transactions of the American Nuclear Society. 2015;112:121-4.
7. J. S. Kim, J. Chen J, H. E. Garcia. "Modeling, control, and dynamic performance analysis of a reverse osmosis desalination plant integrated within hybrid energy systems," Energy. 2016;112:52-66.
8. C. Rabiti, A. Alfonsi, J. Cogliati, D. Mandelli, R. Kinoshita, S. Sen, C. Wang, J. Chen, "RAVEN User Manual," INL/EXT-15-34123, Printed March 2017.
9. J. Chen, C. Rabiti, "Synthetic wind speed scenarios generation for probabilistic analysis of hybrid energy systems", Energy 120, 2017, p. 507-517.
10. P. Talbot. C. Rabiti, A. Alfonsi, C. Krome, M. R. Kunz, A. Epiney, C. Wang, D. Mandelli, "Correlated Synthetic Time Series Generation using Fourier and ARMA", International Journal of Energy Research, accepted 18 October 2019.
11. N. Stauff, G. Maronati, R. Ponciroli, F. Ganda, T. Kim, T. Taiwo, A. Cuadra, M. Todosow, P. Talbot, C. Rabiti, B. Dixon, S. Kim, "Daily Market Capability and Results", ANL/NSE-19/5, April 2019.
12. A. S. Epiney, C. Rabiti, A. Alfonsi, P. Talbot, F. Ganda, "Report on the Economic Optimization of a Demonstration Case for a Static N-R HES Configuration using RAVEN," Idaho National Laboratory, April 2017, INL/EXT-17-41915.
13. Nicolas E. Stauff, R. Ponciroli, T. K. Kim, T. A. Taiwo, "Economic Impact of Flexible Nuclear Operation Estimated with EDGAR Optimization Code," NURER 2018, Sept 30 – Oct 3, 2018, Jeju, Korea.
14. S. M. Cohen, J. Becker, D. A. Bielen, M. Brown, W. J. Cole, K. P. Eurek, A. Frazier, et al, "Regional Energy Deployment System (ReEDS) Model Documentation: Version 2018", National Renewable Energy Laboratory, NREL/TP-6A20-72023, 2019.
15. G. van Rossum, "Python tutorial", Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995.
16. K. Frick, P. Talbot, D. Wendt, R. Boardman, C. Rabiti, S. Bragg-Sitton, D. Levie, B. Frew, M. Ruth, A. Elgowainy, T. Hawkins, "Evaluation of Hydrogen Production Feasibility for a Light Water Reactor in the Midwest," Idaho National Laboratory, September 2019, INL/EXT-19-55395.

17. https://about.gitlab.com, “Code, test, and deploy together with GitLab open source git repo management software”, Retrieved 2016-Februrary-22
18. PJM, “Energy Market”, online resource, accessed at https://www.pjm.com/markets-and-operations/energy.aspx, Retrieved 2019-12-13.

8. APPENDIX: Software Requirements Specification and Traceability Matrix

MANUAL

SPC-xxxx

Revision 0

Printed December 19, 2019

HERON Software Requirements Specification and Traceability Matrix

Paul Talbot, Abhinav Gairola, Prerna Prateek, Andrea Alfonsi, Cristian Rabiti

Prepared by
Idaho National Laboratory
Idaho Falls, Idaho 83415

The Idaho National Laboratory is a multiprogram laboratory operated by
Battelle Energy Alliance for the United States Department of Energy
under DOE Idaho Operations Office. Contract DE-AC07-05ID14517.

Issued by the Idaho National Laboratory, operated for the United States Department of Energy by Battelle Energy Alliance.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

SPC-
xxxx
Revision 0

Printed December 19, 2019

HERON Software Requirements Specification and Traceability Matrix

Paul Talbot, Abhinav Gairola, Prerna Prateek, Andrea Alfonsi, Cristian Rabiti

Contents

1 Introduction

The ***HERON*** plug-in is a generic tool for techno-economic analysis of resource distribution system in Regulated and Deregulated Market.

The plug-in is aimed to assist in component sizing optimization using stochastic optimization of resource allocation.

This document is aimed to report and explain the ***HERON*** software requirements.

1.1 Dependencies and Limitations

The plug-in should be designed with the fewest possible constraints. Ideally the plug-in (in conjunction with RAVEN) should run on a wide variety of evolving hardware, so it should follow well-adopted standards and guidelines. The software should run on any POSIX compliant system (including Windows POSIX emulators such as MinGW)l.

In order to be functional, ***HERON*** depends on the following software/libraries.

- RAVEN (`raven.inl.gov`) and all its dependencies (listed in "RAVEN Software Design Description" - SDD-513)
- Python dill-0.3.1.1

2 References

- ASME NQA 1 2008 with the NQA-1a-2009 addenda, “Quality Assurance Requirements for Nuclear Facility Applications,” First Edition, August 31, 2009.
- ISO/IEC/IEEE 24765:2010(E), “Systems and software engineering Vocabulary,” First Edition, December 15, 2010.
- LWP 13620, “Managing Information Technology Assets”
- SDD-513, “ RAVEN Software Design Description ”
- PLN-5552, “ RAVEN and RAVEN Plug-ins Software Quality Assurance and Maintenance and Operations Plan ”

3 Definitions and Acronyms

3.1 Definitions

- **Baseline.** A specification or product (e.g., project plan, maintenance and operations [M&O] plan, requirements, or design) that has been formally reviewed and agreed upon, that thereafter serves as the basis for use and further development, and that can be changed only by using an approved change control process. [ASME NQA-1-2008 with the NQA-1a-2009 addenda edited]
- **Validation.** Confirmation, through the provision of objective evidence (e.g., acceptance test), that the requirements for a specific intended use or application have been fulfilled. [ISO/IEC/IEEE 24765:2010(E) edited]
- **Verification.**
  - The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.
  - Formal proof of program correctness (e.g., requirements, design, implementation reviews, system tests). [ISO/IEC/IEEE 24765:2010(E) edited]

3.2 Acronyms

API Application Programming Interfaces

ASME American Society of Mechanical Engineers

CR Change Request

DOE Department of Energy

HDF5 Hierarchical Data Format (5)

LWRS Light Water Reactor Sustainability

NEAMS Nuclear Energy Advanced Modeling and Simulation

IES Integrated Energy Systems

INL Idaho National Laboratory

IT Information Technology

M&O Maintenance and Operations

NQA Nuclear Quality Assurance

POSIX Portable Operating System Interface

PP Post-Processor

QA Quality Assurance

RAVEN Risk Analysis and Virtual ENviroment

ROM Reduced Order Model

SDD System Design Description

XML eXtensible Markup Language

4 System Requirements: HERON

4.1 Minimum Requirements

4.1.1 Minimum Requirements

4.1.1.1 H-M-1

Computer: Any POSIX (and POSIX-like) system

4.1.1.2 H-M-2

RAM: 2 GB per core execution (depending on the type of analysis and data genarated)

4.1.1.3 H-M-3

Language: Python 3.x

4.2 Functional Requirements

4.2.1 Framework, I/O, Execution Control

4.2.1.1 H-F-1

The HERON plug-in shall allow support for user-defined instructions for controlling the execution stages of the economical analysis.

4.2.1.2 H-F-2

The HERON plug-in shall allow for user-defined output types for simulation data.

4.3 Usability Requirements

4.3.1 Economical Analysis

4.3.1.1 H-EA-1

The HERON plug-in shall support the automated RAVEN workflow generation for grid energy system technoeconomic analysis

4.3.1.2 H-EA-2

The HERON plug-in shall support the grid system dispatch optimization on stochastic signals

4.3.1.3 H-EA-3

The HERON plug-in shall support the regulated and deregulated market analysis

4.4 System Interfaces

4.4.1 Interface with external applications

4.4.1.1 H-SI-1

The HERON plug-in shall be able to be initialized via input files.

4.5 System Operations

4.5.1 Human System Integration Requirements

The command line interface shall support the ability to toggle any supported coloring schemes on or off pursuant to section 508 of the Rehabilitation Act of 1973.

4.5.2 Maintainability

- The latest working version (defined as the version that passes all tests in the current regression test suite) shall be available at all times through the repository host provider.
- Flaws identified in the system shall be reported and tracked in a ticket or issue based system. The technical lead or any COB member will determine the severity and priority of all reported issues. The technical lead will assign resources at his or her discretion to resolve identified issues.
- The software maintainers will entertain all proposed changes to the system in a timely manner (within two business days).
- The HERON plug-in in its entirety will be made available for licensees in the INL GITLAB system.

4.5.3 Human System Integration Requirements

The regression test suite will cover at least 80% of all lines of code at all times. The results of the regression tests will be stored in the Continuous Integration System.

4.6 Information Management

The HERON plug-in in its entirety will be made available on an appropriate protected repository hosting site (i.e. INL GITLAB). Backups and security services will be provided by the hosting service.

5 Verification

The regression test suite shall employ several verification tests using comparison against analytic solutions (when possible) and convergence rate analysis.

6 HERON:SYSTEM REQUIREMENTS

6.1 Requirements Traceability Matrix

This section contains all of the requirements, requirements' description, and requirement test cases. The requirement tests are automatically tested for each CR (Change Request) by the CIS (Contin- uous Integration System).

6.1.1 Minimum Requirements

Requirment ID	Requirment Description	Test(s)
H-M-1	Computer: Any POSIX (and POSIX-like) system	1)"RAVEN User Manual", INL/EXT-15-34123 2)Continous Integration System
H-M-2	RAM: 2 GB per core execution (depending on the type of analysis and data genarated)	1)"RAVEN User Manual", INL/EXT-15-34123 2)Continous Integration System
H-M-3	Language: Python 3.x	1)"RAVEN User Manual", INL/EXT-15-34123 2)Continous Integration System

Minimum Requirements

6.1.2 Functional Requirements

Requirment ID	Requirment Description	Test(s)
H-F-1	The HERON plug-in shall allow support for user-defined instructions for controlling the execution stages of the economical analysis.	1)/Users/talbpw/projects/heron/tests/-First/myFirst.xml

H-F-2	The HERON plug-in shall allow for user-defined output types for simulation data.	1)/Users/talbpw/projects/heron/tests/-First/myFirst.xml

Framework, I/O, Execution Control

6.1.3 Usability Requirements

Requirment ID	Requirment Description	Test(s)
H-EA-1	The HERON plug-in shall support the automated RAVEN workflow generation for grid energy system technoeconomic analysis	1)/Users/talbpw/projects/heron/tests/-First/myFirst.xml
H-EA-2	The HERON plug-in shall support the grid system dispatch optimization on stochastic signals	1)/Users/talbpw/projects/heron/tests/Second/mySecond.xml
H-EA-3	The HERON plug-in shall support the regulated and deregulated market analysis	1)/Users/talbpw/projects/heron/tests/Third/myThird_1.xml

Economical Analysis

6.1.4 System Interfaces

Requirment ID	Requirment Description	Test(s)
H-SI-1	The HERON plug-in shall be able to be initialized via input files.	1)/Users/talbpw/projects/heron/tests/-First/myFirst.xml

Interface with external applications


INL
aho National laboratory

9. APPENDIX: Configuration Items List

List

HERON Configuration Items List

The INL is a U.S. Department of Energy National Laboratory
operated by Battelle Energy Alliance.

Applicability:	Configuration Items List		eCR Number:

Manual:

REVISION LOG

Rev.	Date	Affected Pages	Revision Description
0	07/31/2019	All	Creation of the Configuration Items List

CONTENTS

10. PURPOSE

This document identifies all HERON plug-in *configuration items (CIs)* (see def.). This document also identifies the level designation needed to modify CIs that can potentially affect the ability of HERON plug-in to comply with NQA-1.

11. SCOPE

This list is intended to identify all CIs for HERON plug-in, to provide a document to submit into the EA repository, and an aid to identify how severe a change to HERON plug-in will be.

12. RESPONSIBILITIES

The Asset Owner is responsible for maintaining this list and, when necessary, updating the EA repository when the configuration items list changes.

The Asset Owner is also responsible for maintaining configuration management in accordance with PLN-5552, "RAVEN and RAVEN Plug-ins Software Quality Assurance Plan."

13. LIST

13.1 Software, Hardware, and Documentation

Table 1. Software, Hardware, and Documentation

Configuration Item	Component	Description	Repository/Location
Application Source Code	HERON plug-in	Source Code for HERON plug-in	GITLAB
	RAVEN	Source Code for the RAVEN code. The HERON requires RAVEN to be functional.	GITHUB
System Software	Python 3.x	HERON Software language ***(Current versions are maintained in the EA repository)***	Enterprise Architecture (EA) (put code here xxxxx)
	Unix-compatible systems	Any compatible Unix system (or Unix-like)	N/A
Support Software	GITLAB CI	Continuous Integration, Verification, Enhancement, and Testing. This is the continuous integration system used by HERON for automatic testing.	GITLAB Installed in all the Regression Automatic Test Machines (Test Servers)
Hardware	Test Servers	These servers are used to test the HERON functionality. It will use a "snapshot" of live data to perform the tests. If testing on the server fails, that version of HERON is sent back to the Development Server for further configuration. ***(Complete and up-to-date list of servers is maintained in EA repository)***	General Purpose Enclave EROB
	Workstations Laptops Personal Computer	These consist of computer terminals that the end users use to access the software.	N/A
Documentation	SDD-xxxxxx	HERON Safety Software Determination	EDMS
	ALL-000853	HERON Quality Level Determination	EDMS
	Entry #xxxxx	HERON Enterprise Architecture	EDMS
	PLN-5552	RAVEN and RAVEN Plug-ins Software Quality Assurance Plan	EDMS
	PLN-5552	RAVEN and RAVEN Plug-ins Configuration Management Plan	EDMS
	PLN-5552	RAVEN Software Test Plan and V&V	EDMS
	PLN-5552	RAVEN and RAVEN Plug-ins Asset Maintenance Plan	EDMS
	SPC-2366	RAVEN Software Requirements Specification and Traceability Matrix	EDMS

	SDD-513	RAVEN Software Design Description	EDMS
	SPC-xxxx	HERON Software Requirements Specification and Traceability Matrix	EDMS
	SDD-513	HERON Software Design Description	EDMS
	INL/EXT-15-34123	RAVEN User Manual	GITHUB
	INL/EXT-18-44465	RAVEN Theory Manual	GITHUB
	INL/EXT-16-38178	RAVEN User Guide	GITHUB
	INL/EXT-xx-xxxxx	HERON User Manual	GITLAB