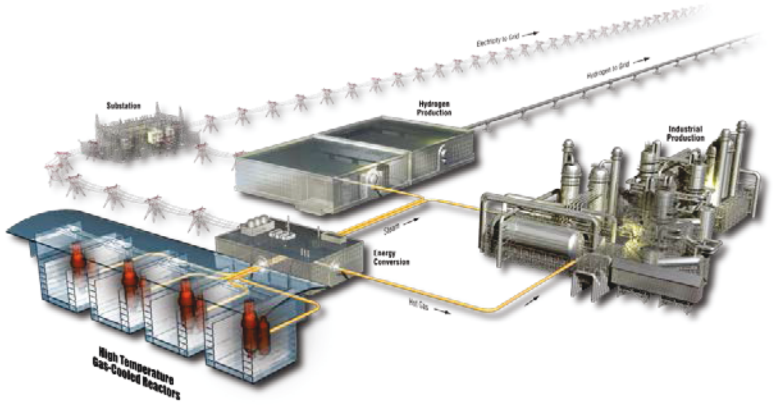
Status on the Development of a Modeling and Simulation Framework for the Economic Assessment of Nuclear Hybrid Energy Systems

Cristian Rabiti
Robert A. Kinoshita
Jong Suk Kim
Wesley Deason
Shannon M. Bragg-Sitton
Richard D. Boardman
Humberto E. Garcia

September 2015



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

Idaho National Laboratory

DISCLAIMER

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

INL/EXT-15-36451
Revision 0

Status on the Development of a Modeling and Simulation Framework for the Economic Assessment of Nuclear Hybrid Energy Systems

Cristian Rabiti
Robert A. Kinoshita
Jong Suk Kim
Wesley Deason
Shannon M. Bragg-Sitton
Richard D. Boardman
Humberto E. Garcia

September 2015

Idaho National Laboratory
INL ART Program
Idaho Falls, Idaho 83415

http://www.inl.gov

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

INL ART Program

Status on the Development of a Modeling and Simulation Framework for the Economic Assessment of Nuclear Hybrid Energy Systems

INL/EXT-15-36451
Revision 0

September 2015

Author:

9/11/2015

Cristian Rabiti
Task Lead
Date

Approved by:

Shannon M Bragg-Sitton
9/9/2015

Shannon M. Bragg-Sitton
Nuclear Hybrid Energy System Co-Lead
Date

Richard D Boardman
9/9/2015

Richard D. Boardman
Nuclear Hybrid Energy System Co-Lead
Date

9/9/2015

Travis R. Mitchell
INL ART TDO Relationship Manager
Date

9/10/2015

Daren K. Jensen
INL ART TDO Quality Assurance
Date

ABSTRACT

An effort to design and build a modeling and simulation framework to assess the economic viability of Nuclear Hybrid Energy Systems (NHES) was undertaken in fiscal year (FY) 2015. The purpose of this report is to document the various tasks associated with the development of such a framework and to provide a status of their progress.

Several tasks have been accomplished. First, starting from the simulation strategy highlighted in [1], a rigorous mathematical formulation has been achieved in which the economic optimization of a Nuclear Hybrid Energy System is presented as a constrained robust (under uncertainty) optimization problem.

Some possible algorithms for the solution of the optimization problem are presented. A variation of the Simultaneous Perturbation Stochastic Approximation algorithm has been implemented in RAVEN, and preliminary tests have been performed.

The development of the software infrastructure to support the simulation of the whole NHES has also moved forward. The coupling between RAVEN and an implementation of the Modelica language (OpenModelica) has been implemented, migrated under several operating systems and tested using an adapted model of a desalination plant. In particular, this exercise focused on testing the coupling of the different code systems; testing parallel, computationally expensive simulations on the INL cluster; and providing a proof of concept for the possibility of using surrogate models to represent the different NHES subsystems. Another important step was the porting of the RAVEN code under the Windows™ operating system. This accomplishment makes RAVEN compatible with the development environment that is being used for dynamic simulation of NHES components.

A very simplified model of the performance of an NHES in the electric market has been built in RAVEN to confirm expectations on the analysis capability of RAVEN to provide insight into system economics and to test the capability of RAVEN to identify limit surfaces even for stochastic constraints. This capability will be needed in the future to enforce the stochastic constraints on the electric demand coverage from the NHES.

The development team gained experience with many of the tools that are currently envisioned for use in the economic analysis of NHES and completed several important steps. Given the complexity of the project, preference has been given to a structural approach in which several independent efforts have been used to build the cornerstone of the simulation framework. While this is a good approach in establishing such a complex framework, it may delay reaching more complete results on the performance of analyzed system configurations.

The integration of the previously reported exergy analysis approach was initially proposed as part of this effort. However, in reality, the exergy-based apportioning of cost will take place only in a second stage of the implementation, since it will be used to properly allocate cost among the different NHES subsystems. Therefore, exergy does not appear at the same level as the main

drivers in the analysis framework; the latter development of the base framework is the focus of this report.

CONTENTS

ABSTRACT vii

ACRONYMS xiii

1. INTRODUCTION 1

2. FROM THE MATHEMATICAL PROBLEM TO SOFTWARE DESIGN 1
   - 2.1 Mathematical Formulation 1
   - 2.2 Overall Software Design 4

3. DRIVER DESIGN AND INITIAL TESTING 7
   - 3.1 Design 7
     - 3.1.1 Summary of Theory 7
   - 3.2 Translation to the Case of Interest 9
   - 3.3 Testing 11
     - 3.3.1 Testing Function 11
     - 3.3.2 Preliminary Results 11

4. COUPLING RAVEN AND MODELICA 13
   - 4.1 Introduction 13
   - 4.2 OpenModelica Interface for RAVEN 14
     - 4.2.1 RAVEN OpenModelica Interface 14
     - 4.2.2 Function generateCommand() 15
     - 4.2.3 Function createNewInput() 15
     - 4.2.4 Function finalizeCodeOutput() 16
     - 4.2.5 Function checkInputExtension() 16
   - 4.3 Modeling and Simulation of Regional NHES Configurations in Modelica 16
   - 4.4 Reverse Osmosis Desalination 18
   - 4.5 Conversion of the RO Desalination Plant Model to OpenModelica 19
   - 4.6 Running the Model with RAVEN 22
     - 4.6.1 Generation of the OpenModelica Executable Files 22
     - 4.6.2 Simple RAVEN Input Structure 23
   - 4.7 Limit Surface Search 23
   - 4.8 Surrogate Model Test 25

5. DEMONSTRATION OF THE ECONOMIC MODEL 28
   - 5.1 Model Description 28
   - 5.2 Results Discussion 32
     - 5.2.1 Parametric Studies 32
     - 5.2.2 Reliability Surface Analysis 34

6. ENHANCEMENT OF THE DEVELOPMENT FRAMEWORK: RUNNING RAVEN UNDER WINDOWS™ OS 36

7. FUTURE ACTIVITIES 36

8. CONCLUSIONS 37

9. REFERENCES 38

FIGURES

Figure 1. Generalization of possible NHES components and boundaries. 2

Figure 2. Demand (January 1 to April 30, 2015) in the ERCOT region from http://www.ercot.com/gridinfo/load/load_hist/. 3

Figure 3. Simulation framework. 5

Figure 4. Simplified simulation framework. 6

Figure 5. Algorithm flowchart for the constrained robust optimization. 10

Figure 8. OpenModelica tools. 14

Figure 9. NHES option in West Texas implemented in Modelica/Dymola. 17

Figure 10. NHES option in Arizona implemented in Modelica/Dymola. 18

Figure 11. RO desalination: (a) process flow diagram for a two-stage BWRO plant and (b) a schematic of an RO vessel, which consists of six membrane modules in series [2,3]. 19

Figure 12. RO desalination plant model in Modelica/OpenModelica. 21

Figure 13. Command for the creation of the OpenModelica executable. 22

Figure 14. Limit surface search algorithm. 24

Figure 15. Limit surface. 25

Figure 16. Flow chart for the economic assessment demonstration tool. 31

Figure 17. LCOE parametric studies (Total cost is equivalent to LCOE_Total). 32

Figure 18. Total renewable capacity installed. 33

Figure 19. Hours for which demand cannot be met over a one-year period. 33

Figure 20. LCOE parametric studies 34

Figure 21. Limit surface illustrating the parametric region where the system will not lead to more than 30 hours of loss of load over a year time frame. 35

TABLES

Table 1. Code interface methods 14

Table 2. Potential NHES configurations. 16

Table 3. RO desalination model inputs. 20

Table 4. RO desalination model outputs. 20

Table 5. Modelica files used in an RO desalination plant 21

Table 6. Input parameters (nominal and bound value) for the RO plant. 26

Table 7. Data set descriptions used for creation of the surrogate models and their cross -validation... 26

Table 8. Comparison of ROMs versus original model (training set: Monte Carlo 1, cross-validation set Monte Carlo 1)... 27

Table 9. Comparison of ROMs vs. original model (training set: Monte Carlo 1, cross validation set Monte Carlo 2). 27

Table 10. Comparison of ROMs vs. original model (training set: Grid, cross validation set Monte Carlo 2). 28

ACRONYMS

ACE	Area Control Error
API	application programming interface
ART	Advanced Reactor Technologies
BWRO	Brackish Water Reverse Osmosis
CDF	Cumulative Distribution Function
CSV	Comma Separated Values
DOE	U.S. Department of Energy
Dymola	Dynamic modeling library
ERCOT	Electric Reliability Council of Texas
FMI	Functional Mockup Interfaces
FMU	Functional Mockup Unit
FOM	figure of merit
HP	high-pressure
HPC	high performance computing
INL	Idaho National Laboratory
LOLP	Loss of Load Probability
MGPD	million gallons per day
NGTL	Natural Gas-To-Liquid
NHES	Nuclear hybrid energy systems
NREL	National Renewable Energy Laboratory
OMEdit	OpenModelica Connection Editor Analysis Package
PV	Photovoltaic
RO	reverse osmosis
SDG	Stochastic Descent Gradient
SPSA	Perturbation Stochastic Approximation
TDS	Total Dissolved Solids
XML	eXtensible Markup Language

Status on the Development of a Modeling and Simulation Framework for the Economic Assessment of Nuclear Hybrid Energy Systems

1. INTRODUCTION

This report provides the current status of development of a software framework to perform economic performance evaluation of NHES (Nuclear Hybrid Energy Systems). This document first illustrates the mathematical formulation of the problem and a selected set of options that are being tested and/or discussed concerning which software/library should be used for the analysis and simulations. The report then highlights the fiscal year (FY) 2015 developments, provides the status of the defined tasks, and defines a development plan for the accomplishment of the economic assessment of NHES.

2. FROM THE MATHEMATICAL PROBLEM TO SOFTWARE DESIGN

2.1 Mathematical Formulation

As largely discussed in Rabiti et al. 2015 report [1], the problem to solve is to find the optimal configuration of an NHES that will minimize the cost of electricity production, while accounting for defined constraints on the capability of the NHES to meet demand. These constraints have a fundamental role in enabling the economic evaluation framework by monetizing the ability of the NHES being analyzed to better cope with electricity demand volatility. Note that if variable renewable resources are present, either within the NHES or connected to the grid external to the NHES, then it is the volatility in the net demand (total demand less the renewable contribution) that is addressed.

While it is possible to analyze the problem by maximizing profit, minimization of the cost of the electricity production is alternatively chosen here as a reference approach. The profit maximization approach requires an electricity price, and is therefore defined within a specific market. The current markets are not designed to minimize electricity cost, but to minimize the electricity production marginal cost (suppliers are willing to accept any price above their marginal costs). The minimization of production marginal cost, in the short run, ensures lower cost to the grid and to users; however, it does not ensure, in the long run, that the suppliers with the lowest total cost (or economical cost) will stay in the market. If those suppliers, usually characterized by low marginal cost and high fixed costs, are forced to leave the market due to an inability to recover the capital costs over an extended operating time, the price of electricity will have an upward trend that could be fairly steep. This trend will be characterized by an energy portfolio drifting toward a supplier mix that is more dominated by high marginal cost, but generally low capital cost suppliers. Although these suppliers may have higher overall (lifetime) costs, utilities that can recover the marginal costs at the time of use are capable of recovering total costs.

NHES might be among the suppliers with low marginal costs, incapable of recovering capital costs in the current market that subsidizes renewable generators, despite having an economical cost for electricity production that is below that of several of the current suppliers (more precise economical evaluation for NHES are still needed). Therefore, a long-term analysis that is obtained by a cost-based approach seems more suitable to reveal the benefits of the NHES.

Moreover, the high degree of dispatchability of the NHES might be capable of absorbing the volatility of the demand/supply dynamic, allowing more base-load suppliers to stay in the market with the benefit of decreasing the need to use a costly, highly dispatchable electricity supplier such as natural gas turbines. The decrease in volatility allows base-load suppliers to increase their contribution to covering electricity demand. Those suppliers are usually the ones having the lowest marginal costs; therefore, NHES could lower the average total cost of electricity production.

For the above reason, the approach chosen here is cost-based and should include a means to capture the reduction of cost at the system level by decreased need for costly dispatchable resources.

In the cost minimization approach, the "source term" appearing on the right side of the equation set (presented and discussed below) that defines the response of the NHES is the electricity demand or the net demand (demand less the renewable production). Figure 1 shows the possible definition of the NHES boundary. For a fully-coupled hybrid system, the blue box defines the boundary of the NHES, which includes both nuclear and renewable generators. If the boundary condition is redefined such that the nuclear and renewable components are more loosely coupled (orange box in Figure 1), then the generator (the nuclear plant) would need to respond to the net demand from the grid.

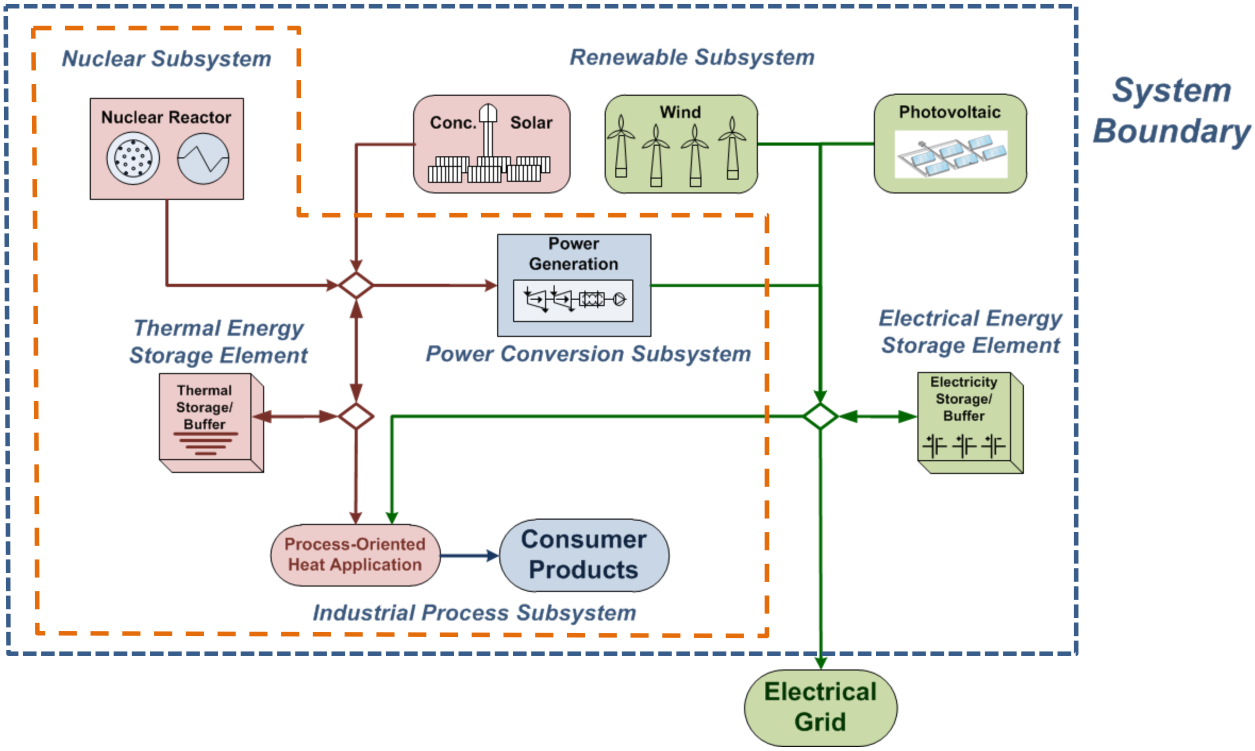

Nuclear Subsystem
Renewable Subsystem
System Boundary
Nuclear Reactor
Conc. Solar
Wind
Photovoltaic
Power Generation
Thermal Energy Storage Element
Power Conversion Subsystem
Electrical Energy Storage Element
Thermal Storage/ Buffer
Electricity Storage/ Buffer
Process-Oriented Heat Application
Consumer Products
Industrial Process Subsystem
Electrical Grid

Figure 1. Generalization of possible NHES components and boundaries.

If it is not required to make an explicit distinction between net demand and demand, the term demand, d, will be used, where d is a function of time and has a stochastic behavior. Generally speaking, demand is always a source term in the set of equations presented below. When it is relevant to highlight how the contribution of the renewable is accounted, the distinction between net demand and demand is relevant. For example, when the system is exposed to demand and renewable generation is present it is necessary to explicitly model the energy supply by the renewable. If net demand is used instead then no direct modeling of the renewable is necessary.

Figure 2 shows the demand in the Electric Reliability Council of Texas (ERCOT) region at the beginning of year 2015 (total).

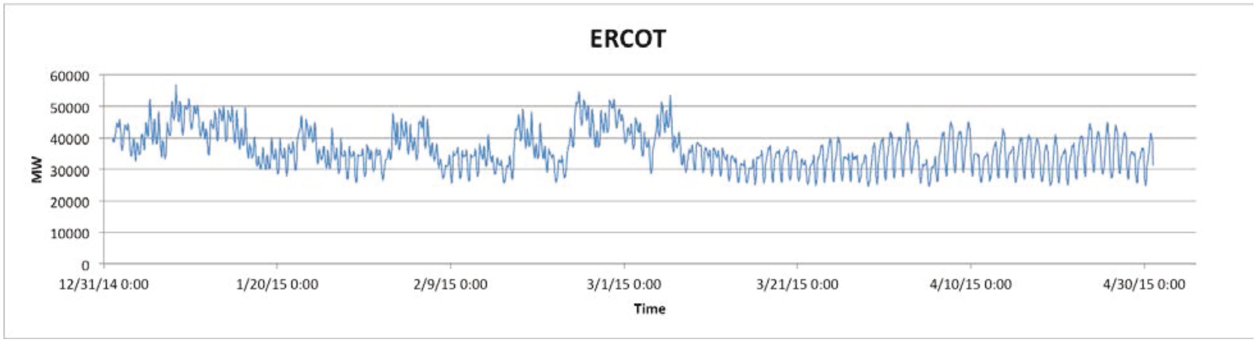


Figure 2. Demand (January 1 to April 30, 2015) in the ERCOT region from http://www.ercot.com/gridinfo/load/load_hist/.

Demand has a stochastic behavior of unknown distribution; therefore, it cannot be modeled analytically. While it would be possible to use real data acquired over several years, this still would not provide any information about the response of NHES to a stochastic demand; this approach will instead just limit the specific realization represented by that time history of demand.

The correct approach is to define a time window (period) and to construct a statistical representation of the demand within that time window. The time window should be large enough so that the system could react to the characteristics variation within that time period. In other words, it is necessary for the period to be long enough that the system can fully adapt to any variation in the demand (bounded by its characteristic statistical dispersion) within the period itself.

For example, recalling that one of the tests suggested to verify the statistical capability of the system to cover demand was based on the Loss of Load Probability (LOLP) [1]. LOLP is tested on an hourly basis; the choice of 1 day as the characteristic period would be appropriate. At this point it would be out of scope to further analyze how to structure the time discretization of the problem; this task will be undertaken in FY16. However, it can be assumed that it is possible to determine d as the realization of the random variate D:

$$d(t) \sim D(t)$$

The definition of a random variate over time is still not an easy concept to represent numerically. It is necessary to define the dimensionality of the random variate to be the number of values used to represent the demand evolution over the period. For the case of an LOLP-type test, the values are clearly the hourly demand within the one day period. In this case, a realization of $D(t)$ will be a vector of values representing the hourly demand over a random day.

To generalize the above consideration, the choice of a day as a reference period and 1 hour as the resolution time is dependent only on the characteristic time of the phenomena to be examined (fluctuation of the demand and response of the NHES. For example, if an Area Control Error (ACE) is chosen as the reliability metric, a reasonable choice for the period could be 1 hour and the resolution interval could be on the order of 1 second.

This report refers to the 1-hour resolution and 1-day period (LOLP-type of test) since the shorter time scale will be of interest when the real-time digital simulator (RTDS) capabilities are integrated in the framework in the future. That is currently an area of future development, and selection of the hourly resolution is in agreement with the capability of Modelica [2] based dynamic models of NHES architectures.

The type of analysis considered thus far would suggest that the output needed from the simulation of the NHES $\boldsymbol{S}$ will be the vector of uncovered demand [kWh], the number of times the demand was uncovered [#], and the cost per electricity unit supplied [$/kWh], respectively $u, n,$ and c:

$$\boldsymbol{S} = (u, n, c)$$

For a given realization d of the demand the NHES representation should return:

$$\boldsymbol{S}(d) = \big(u(d), n(d), c(d)\big)$$

During the design stage there are parameters $\boldsymbol{p}$ that could be changed to improve the performance of the system. Examples of those parameters are the total installed capacity for a given average demand, the fraction of the capacity of each subsystem, the capacity of battery installed, and the design parameters that might control the tradeoff between faster ramp-up time and efficiency of the plant, etc. Those parameters could be incorporated in the above representation of the system:

$$\boldsymbol{S}(d, \boldsymbol{p}) = \big(u(\boldsymbol{p}, d), n(\boldsymbol{p}, d), c(\boldsymbol{p}, d)\big)$$

As already mentioned, the LOLP test could be one of the many metrics to assess the capability of the system (in an hourly timescale) to meet demand. In the formulation used so far, failing to meet demand within a period (1 day) in any of the hours would imply $n > 0$. Since d is a realization of a random variate, then u, n, and c are as well. Consequently, the LOLP test could be formulated as:

$$CDF(n > 0) < LOLP_{toll}$$

This representation states that the Cumulative Distribution Function (CDF) of the number of times the demand was not met should be below an established LOLP threshold. The LOLP threshold is discretionary; a parametric study with respect to such threshold could highlight how the costs increase by defining a stricter constraint on grid reliability (lower LOLP threshold).

The problem can now be fully defined by the solution of the set of equations to determine:

$$c_{min} = min\{c | c = c(\boldsymbol{p}, d) \ \cup \ CDF(n(\boldsymbol{p}, d) > 0) < LOLP_{toll}\}$$

It is expected that the minimization process would not only determine c_{min} but also $\boldsymbol{p}$**,** in particular, by enforcing the constraint on the LOLP. This is not mathematically ensured, possibly even unlikely, since there can be different systems leading to the same minimum cost but having different configurations while still meeting the reliability constraint. If this will happens, it will not impact the economic assessment, but it could impact the system design. Possible counter measures will need to be considered in the future.

2.2 Overall Software Design

The problem, from a software design point of view, can be split in two main blocks: one is the constrained optimization under uncertainty, the other is the physical representation of the system $\boldsymbol{S}(d, \boldsymbol{p})$.

Both problems are challenging, but the physical representation of the different subsystems that might compose the NHES has progressed via the dynamic regional case analysis completed earlier in FY15 [3]. Therefore, it is natural to now look to the optimization problem. Previous work in the NHES project has not explored the optimization problem to a great extent. For this reason, one of the primary tasks for FY15 has focused on the development of the modeling and simulation infrastructure and probing possible approaches to solve the constrained optimization problem.

An additional reason to begin development of the constrained optimization, now referred to as the “driver” or “sampler” as will be illustrated later in Section 3, is that in a risk weighted approach, for the software development plan, this is the highest complexity component with the lowest cost to development. Therefore, it is natural that a feasibility assessment should start by analyzing this problem. Following the rationale already discussed in the previous gap analysis report [1] and given a series of synergies and already available capabilities, the choice to develop such a driver within the Risk Analysis Virtual Environment (RAVEN) framework is the most cost effective.

The following list provides a brief summary of capabilities that are already available in RAVEN that will be needed and leveraged in the driver for the economic assessment of NHES.

- Statistical packages for the generation of random sampling on given distributions
- General sampling strategies for the exploration of stochastic systems from Monte Carlo to reliability surface search
- Large library of surrogate models that could be utilized to accelerate the search for the solution
- Graphical post-processing for the analysis of the data
- Statistical post processing to investigate the property of the system (e.g.. correlation analysis, topological analysis).

Section 3 describes how some of these capabilities could be leveraged and combined to create the optimization driver, along with preliminary tests.

With respect to physical representation of the system, the gap analysis [1] suggested Ptolemy II as the hub for the communication among the physical representation of different subsystems. The sub-systems could eventually be coded in different languages and using the Functional Mock-up Interface (FMI) as a common data exchange protocol [4]. This overall scheme is reported in Figure 3.

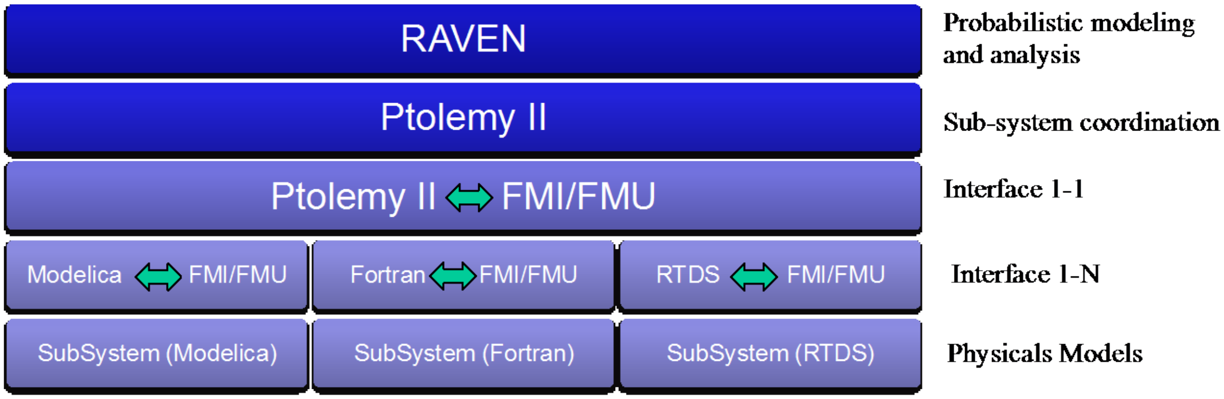


Figure 3. Simulation framework.

From a software engineering point of view, this approach has the advantage that the number of interfaces grows linearly with the number of subsystems (1-N). This is one of the main advantages of the "hub and spoke" approach. At the same time, when the number of subsystems is low, the initial number of interfaces is high compared to the number of subsystems – making this approach ineffective for a small number of subsystems. While the hub and spoke approach based on Ptolemy II (or similar) is indeed the correct long-term approach, at this moment, given the successful experience using Modelica for the dynamic analysis of regional cases [3] and consideration of the number of interfaces for a small number of subsystems, direct coupling between RAVEN and Modelica was explored.

This approach, as will be shown later, allows the investigation of the applicability of RAVEN-based algorithms to the analysis of the subsystems composing the NHES. In particular, it allows starting the evaluation of the use of surrogate models to accelerate the solution of the optimization problem. In this case, the simulation framework is simplified, as shown in Figure 4. Section 4 of this report describes the development of the interface between subsystem models developed in Modelica with RAVEN.

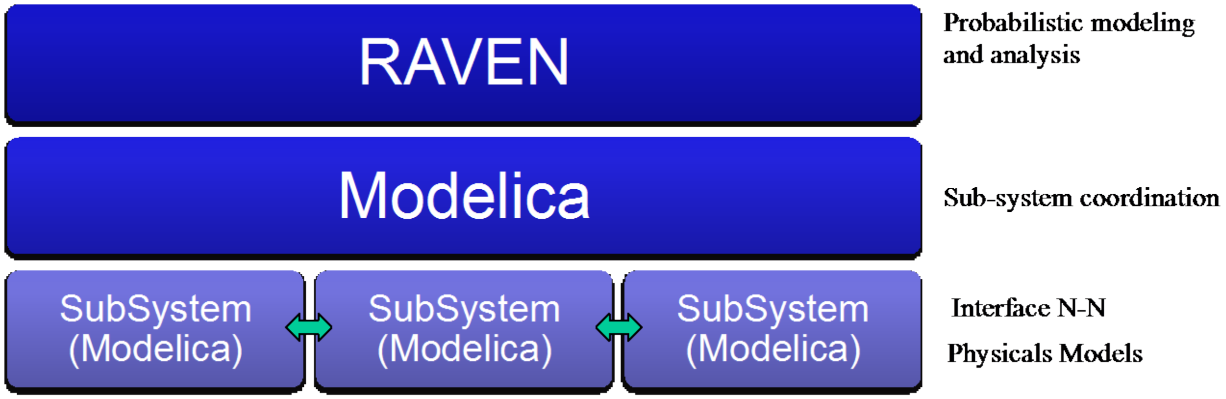


Figure 4. Simplified simulation framework.

Reduced Order Models (ROM), as already mentioned, could be a means to accelerate the search for the set of parameters leading to the minimum electricity cost for a given constraint on the demand coverage. Hence, it is necessary to clarify what a surrogate model is and how it could be used. A surrogate model (e.g., ROM, meta-model, supervised learning) can be characterized as follows:

- Provides a replacement for a physical model when:
  - Only a set of specific figures of merit (FOMs) are considered to characterize the physical system
  - The parameters characterizing the input space are bounded within a certain set of values
- Has a very fast solving time (usually far below a second)
- Can be trained to improve its capability to predict the system FOMs
- Further training increases the computational cost for training and for the evaluation.

RAVEN possesses the capability to sample a physical system represented by a computational model (software) and generate surrogate models. RAVEN has access to all supervised learning algorithms contained in the scikit learn library [5] and a few internally developed algorithms, such as inverse weight, stochastic collocation, n-dimensional splines, and linear topological decomposition [6].

Section 4 of this report illustrates how a ROM of the desalination plant model, programmed in Modelica, can be built and how well it compares to the original model. Section 3 describes how the ROMs can then be used to accelerate the solution of the optimization problem. In stochastic analysis, the system needs to be sampled many times, where several of those samples lead to evaluation of the sub-systems with a practically identical set of input parameters. In such cases, a surrogate of the model representing the sub-system is usually used since there is no need to run the sub-system model once more.

Another issue that appeared during the initial development of the framework was the long-term necessity to have the framework capable of working under several operating systems (Windows™, Linux, and Mac) to allow the team developing the different subsystem models to operate in the native environment for the specific code system selected and where the team already has a development environment set up. This approach also allows for use of though they might be using expensive commercial software that is not portable across platforms. For this reason several components of the simulation framework proposed have been tested and eventually adapted cross-platform. This effort is documented in Section 6.

RAVEN is currently developed at INL under NQA-1 (quality level designation 3) and is currently getting ready to be reclassified as quality level 2. While this indeed generates an extra burden in the development process it ensures full traceability and stability of the code. The Modelica interpreters (OpenModelica and Dymola) such as Ptolemy II are third party software without a NQA-1 classification.

Consequently, as is the practice in such cases, the models built using those tools will be built using NQA-1 accepted processes and will become part of the regression testing suite to ensure coherence of the results over time.

3. DRIVER DESIGN AND INITIAL TESTING

This section is dedicated to the design and preliminary testing of an algorithm that could be used to perform the necessary constrained optimization under uncertainties. It is assumed that the stochastic models for the demand and for the renewable generator are available. These models could be derived either by constructing a stochastic representation of those quantities or by just directly sampling databases.

3.1 Design

The field of robust optimization, and in the specific case of stochastic optimization, has received a lot of attention in recent years [7,8], and this field continues to grow rapidly. Many possible approaches will be briefly discussed here. The 2006 article by Huang, Allen, Notz, and Zeng [8] addresses specific algorithms and provides a reasonable overview of available algorithms for robust optimization.

The objective in the current work is to implement an algorithm that leverages the current RAVEN capabilities to help one to understand how to better characterize the problem and thus provide more effective approaches.

First, consideration that many of the optimization algorithms are based on:

- Sampling of the system.
- Training of a predictive model to determine the location of the extremum.
- Use of an active learning logic to determine the next training point.
- Testing the accuracy of the predictive model. If the convergence test is passed, the algorithm quits; otherwise, the algorithm starts again from the first point.

In the following approach, the predictive model is a local approximation of the gradient, but many other types of ROMs have been used to speed up the optimization as noted by Huang, Allen, Notz, and Zeng [8], where the ROM used is a Gaussian Process (also known as Kriging). For this reason, later sections of this report give particular emphasis to testing the coupling between RAVEN and Modelica-based models to generate different ROMs and to determine limit surfaces. Limit surfaces are the region of the input parameter where specific constraints are satisfied. In the particular case studied here, the limit surface is the region where the reliability constraint is satisfied.

3.1.1 Summary of Theory

The algorithm explored here is based on the Simultaneous Perturbation Stochastic Approximation (SPSA) [9,10]. To briefly sketch the idea behind this algorithm, start by considering a minimization problem as follows:

$$\bar{x}_{min} = \min_{\bar{x}\in R^n} f(\bar{x})$$

$$f(\bar{x}): R^n \to R$$

The standard descent algorithm reads:

$$\bar{x}_{min} = \lim_{k\to\infty} \bar{x}_k$$

$$\bar{x}_{k+1} = \bar{x}_k - \alpha\bar{\nabla}f(\bar{x})|_{\bar{x}_k},$$

where α is the learning rate. When the function to be minimized is stochastic (as in the current case, where the cost of electricity depends on the renewable availability and demand, which are both stochastic), the algorithm can be recast in terms of expected value $E[\quad]$:

$$E[\bar{x}_{min}] = \lim_{k\to\infty} \bar{x}_k$$

$$\bar{x}_{k+1} = \bar{x}_k - \alpha E\left[\bar{\nabla} f(\bar{x})|_{\bar{x}_k}\right]$$

The standard descent algorithm can be replaced by the Stochastic Descent Gradient (SDG) algorithm. The SDG formulation is based on the fact that the expected value of the gradient can be replaced, under some assumption, by a few, or even one, evaluations of the goal function.

$$\bar{x}_{k+1} = \bar{x}_k - \alpha \frac{1}{m} \sum_{i=1}^{m} \bar{\nabla} f_i(\bar{x})|_{\bar{x}_k},$$

where i identifies one of the possible realizations (given $\bar{x}_k$) of the cost function. While this is one of the most-used optimization searches (not heuristic), it still has major drawbacks. When neither the goal function nor its derivative is directly accessible, the evaluation of the gradient is commonly performed using the finite difference approach:

$$\bar{\nabla} f_i(\bar{x})|_{\bar{x}_k} \approx \frac{f_i(\bar{x}+c_k\bar{I}) - f_i(\bar{x}-c_k\bar{I})}{2c_k},$$

where c_k is a small number (with respect the characteristic mathematical scale of the problem) and $\bar{I}$ is the identity vector. The choice of c_k is always controversial. In RAVEN this problem is solved by normalizing each dimension separately and using a variable c_k as in Spall [7].

This described approach requires 2n evaluation of the goal function, which could become very expensive when the number of optimization parameters is on the order of several tens. This problem is overcome by the Simultaneous Perturbation Stochastic Approximation approach where the gradient is approximated. The identity vector $\bar{I}$ is replaced by a random vector $\bar{\mathrm{r}}$ and the gradient is approximated as it follows:

$$\bar{\nabla} f_i(\bar{x})|_{\bar{x}_k} \approx \left(\frac{f_i(\bar{x}+c_k\bar{\mathrm{r}}_i) - f_i(\bar{x}-c_k\bar{\mathrm{r}}_i)}{2c_k\mathrm{r}_{i,1}}, \ldots, \frac{f_i(\bar{x}+c_k\bar{\mathrm{r}}_i) - f_i(\bar{x}-c_k\bar{\mathrm{r}}_i)}{2c_k\mathrm{r}_{i,n}}\right).$$

Methodologies to choose α and c_k are discussed in literature. For example, in Flores' 2000 dissertation [11], one can find a quite extensive review; unfortunately, their optimal choice will be case dependent. Another question is how many samples are necessary to obtain a reasonable estimation of the mean gradient (*m*). This, *m*, is controlled by the ratio between the steepness of the goal function (magnitude of the gradient) and its dispersion (sigma): the larger this ratio is the larger will have *m* to be . This ratio is a local value in the parametric space; therefore, adaptive methodologies should be evaluated to improve the overall performance of the algorithm allowing for variable *m*, α and c_k.

Having discussed the problem of free optimization above, the reliability constraint must now be defined. Constrained robust optimization has been dealt with in previous work but still represents a formidable challenge [12].

The initial approach suggested for the current problem is to:

1. Evaluate the constraint function $g(\bar{x})$ at the next iterate point $\bar{x}_{k+1}$
2. Proceed normally (point 3) if the constraint is satisfied; if not:
    a. Generate a series of random vectors $\bar{\mathrm{r}}$ normal to $\bar{\nabla} f(\bar{x})|_{\bar{x}_k}$ until the constraint is satisfied at $\bar{\mathrm{r}} + \bar{x}$.

    b. Move the vector $\bar{\mathrm{r}}$ toward $\bar{\nabla} f(\bar{x})|_{\bar{x}_k}$ until $\bar{\mathrm{r}}$ is tangent to the limit surface

c. Pose $\bar{x}_{k+1} = \bar{x}_k + \bar{\mathrm{r}}$

3. If the constraint is satisfied, proceed with the normal SPSA algorithm.

This approach is far from being optimized but ensures that the constraint is satisfied and preserves some of the information concerning the steepest direction (gradient).

Other approaches have been suggested using penalty functions based on the Lagrangian formulation of the constraint [12] or sequence of ROMs, as in Huang, Allen, Notz, and Zeng's 2006 journal article [8]. In case the approach suggested in the 2006 journal article [8] is feasible, the constraint can be enforced using the limit surface finding algorithms in RAVEN. This is one of the reasons for performing the test on the search of the limit surface for a stochastic constraint in section 5.

3.2 Translation to the Case of Interest

For the evaluation of NHES, the goal function is the cost of electricity production $u(\boldsymbol{p}, d)$, and the constraint is given by the number of hours in which demand would exceed the available capacity over a certain period of time or translated in probability terms $CDF(n(\boldsymbol{p}, d) > 0) < LOLP_{toll}$.

As already discussed, u is a stochastic variable; therefore, its minimization with respect the parameters $\boldsymbol{p}$ could be performed using the SPSA algorithm. In principle, the reliability constraint could deal with the algorithm described above, where the limit surface is the hyper-surface in the parametric space separating the region where the constraint is satisfied and where it is not.

The problem with this type of constraint is that it is a threshold on a statistical quantity having a very small dispersion, and the threshold of interest lies in the tail of the distribution. For example, if a period of 1 day is considered at the resolution time of 1 hour, and the threshold (constraint) is posed as a 1 hour loss of load period over 10,000 hours, then the likelihood of detecting the failure when testing a random day is very low.

This scenario would push the number of sub-cycles to be fairly high, ideally realizing about 10,000 hours within each sub-cycle (m summation upper boundary). This is one of the reasons why statistical analysis of LOLP is very difficult. This problem can be circumvented, as illustrated in the NERC 2011 report [13], by proper selection of the days to be tested (worst scenario day selection). This approach can be seen as a biased Monte Carlo or a variance reduction methodology. Another solution could be to replace the system model with a surrogate using active learning techniques. At present, this work has not advanced so far to establish a valuable acceleration methodology; this task will be addressed in FY16.

The algorithm flow derived and implemented to date is reported in Figure 5..

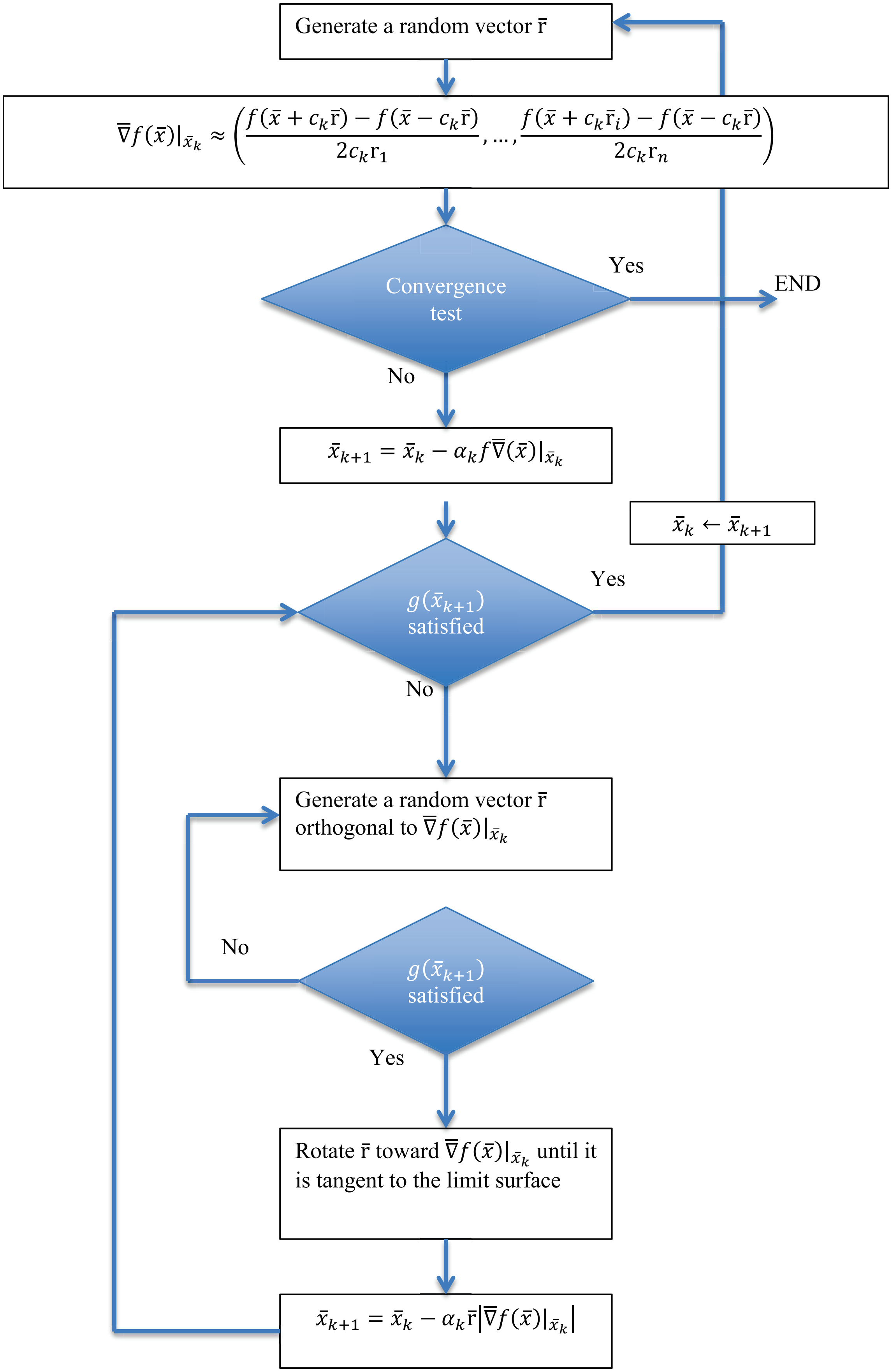

Generate a random vector $\bar{\mathrm{r}}$
$\bar{\nabla} f(\bar{x})|_{\bar{x}_k} \approx \left(\frac{f(\bar{x}+c_k\bar{\mathrm{r}})-f(\bar{x}-c_k\bar{\mathrm{r}})}{2c_k\mathrm{r}_1}, \dots, \frac{f(\bar{x}+c_k\bar{\mathrm{r}}_i)-f(\bar{x}-c_k\bar{\mathrm{r}})}{2c_k\mathrm{r}_n}\right)$
Convergence test
Yes
END
No
$\bar{x}_{k+1} = \bar{x}_k - \alpha_k f\bar{\nabla}(\bar{x})|_{\bar{x}_k}$
$\bar{x}_k \leftarrow \bar{x}_{k+1}$
Yes
$g(\bar{x}_{k+1})$ satisfied
No
Generate a random vector $\bar{\mathrm{r}}$ orthogonal to $\bar{\nabla} f(\bar{x})|_{\bar{x}_k}$
No
$g(\bar{x}_{k+1})$ satisfied
Yes
Rotate $\bar{\mathrm{r}}$ toward $\bar{\nabla} f(\bar{x})|_{\bar{x}_k}$ until it is tangent to the limit surface
$\bar{x}_{k+1} = \bar{x}_k - \alpha_k \bar{\mathrm{r}}\left|\bar{\nabla} f(\bar{x})|_{\bar{x}_k}\right|$

Figure 5. Algorithm flowchart for the constrained robust optimization.

3.3 Testing

3.3.1 Testing Function

The testing function is a mathematical problem that presents the same characteristics of the final problem to solve:

Cost function: $c = F_B B + V_B(d - R * f) + F_R R$

Constraint: $B + R * f > d$

In the above equations the availability factor f and the demand d are normally distributed (the distributions reported here are then truncated between [0,1] and properly renormalized):

$$f \approx d \approx N(\mu = 0.5, \sigma = 0.05)$$

F_b and F_R represent, respectively, the unit fixed cost for a unit of installed capacity of base load and variable renewable (e.g., wind, solar). The optimization parameters are the installed capacity of base load B and of renewable R.

3.3.2 Preliminary Results

First, the nominal solution of the problem can be analytically computed choosing the following values for the parameters:

$F_B = 0.7$

$V_B = 0.3$

$F_R = 1$

The constraint expressed for the mean value reads $B + R * 0.5 > 0.5$, and the gradient of the cost function is $\overline{\nabla} c = (0.7, 0.85)$. The solution of the problem is:

$$R = N(\mu = 0, \sigma = 0.070)$$

$$B = N(\mu = 0.5, \sigma = 0.070)$$

The solution obtained numerically is R = 0.0000683 and B = 0.524194568, which are both within one sigma of the theoretical results. Figure 6 reports the convergence history of the algorithm, while Figure 7 shows the searching pattern in the 3D space (R,B,c).

While this example is only a test, the chosen algorithm and its implementation have shown positive results. In the future more investigation will be needed to assess the robustness of the algorithm and its performance.

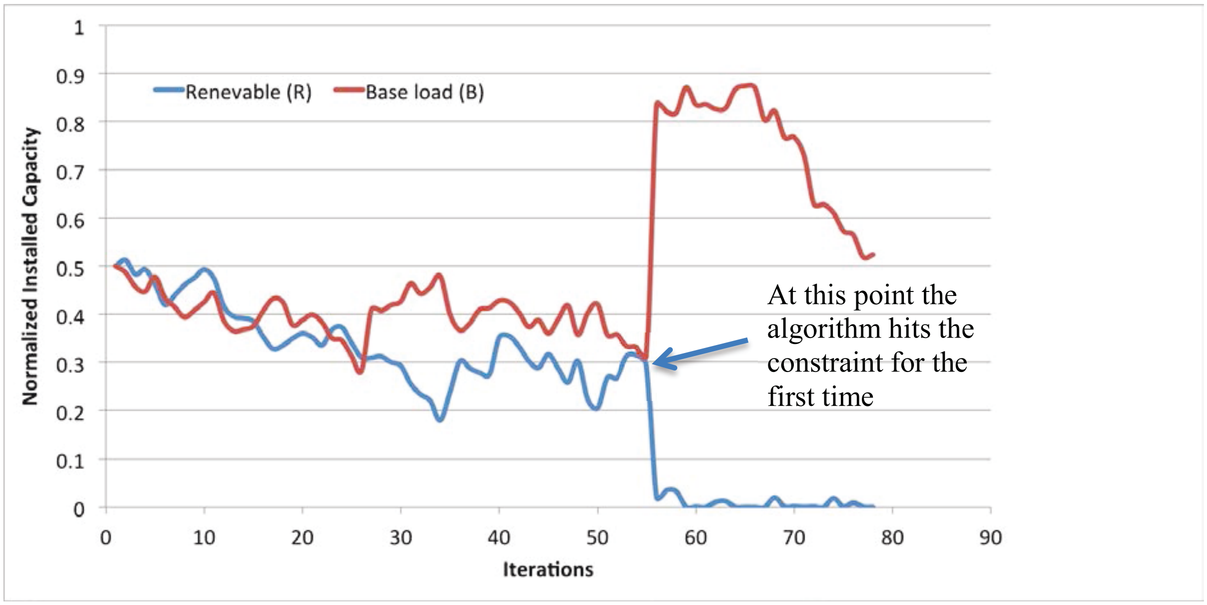


Figure 6. Convergence history.

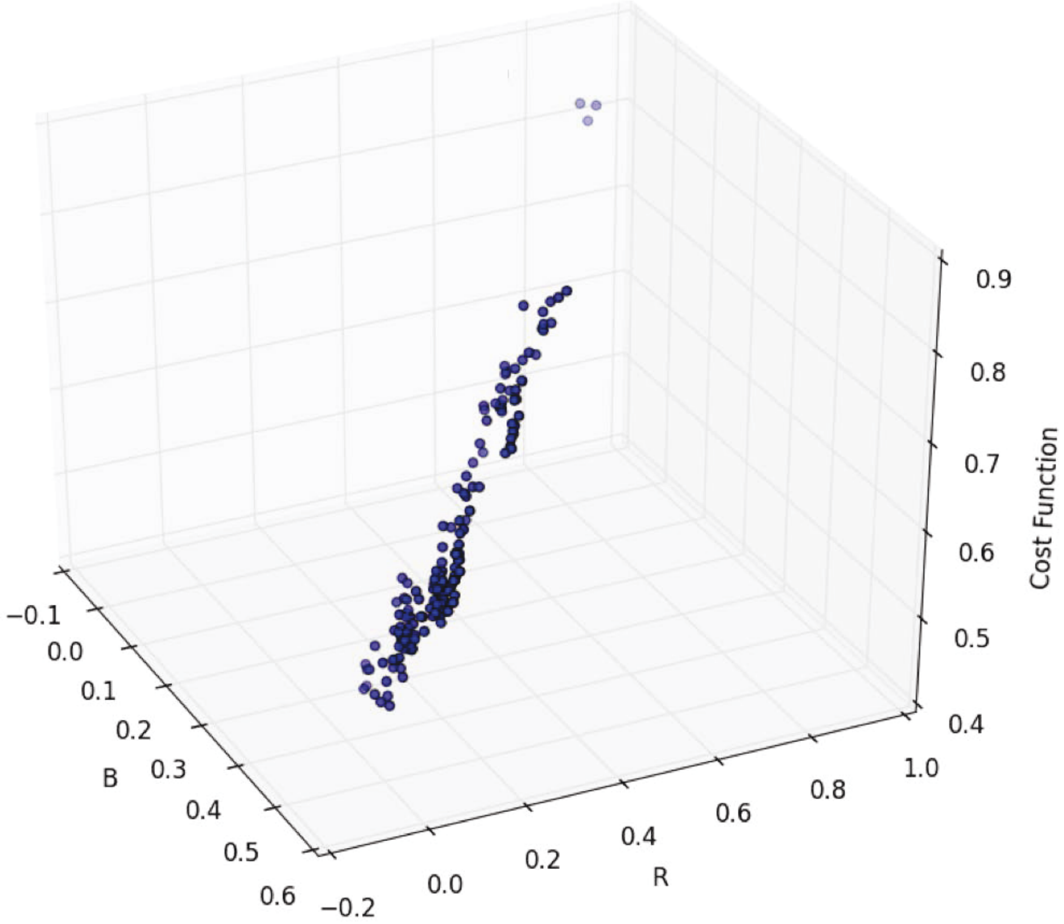


Figure 7: Searching pattern.

4. COUPLING RAVEN AND MODELICA

4.1 Introduction

The study of Hybrid Energy Systems at Idaho National Laboratory (INL) and in the larger Department of Energy (DOE) complex has resulted in the creation of complex models of physical systems. Many of these models have been created using the Dynamic Modeling Library (Dymola) implementation of the Modelica modeling and simulation language. It is desired to use RAVEN with these models to conduct multi-variate parametric studies. RAVEN's flexible nature allows it to be easily interfaced to external models of various types. For this work, an interface to one of the previously developed Modelica implementations was created.

NHES development work at INL is primarily accomplished using Dymola, which is a commercial product of the Dassault Systemes Company. Ideally, these models would be used directly by RAVEN to perform the studies. It would be straightforward to create a RAVEN interface; however, considerations of platform and licensing caused the team to consider other options for coupling in the near term.

The Dymola development environment is only available for Windows and Linux platforms. The INL NHES team currently uses only the Windows version, which typically runs models singly on desktop computers. A RAVEN-based parametric study requires many simulation runs, and as such supports the use of High Performance Computing (HPC). INL has HPC resources available that use the Linux operating system.[a] Therefore, running a Dymola model under RAVEN on the INL HPC platform would require a Linux executable of that model.

These executable models, created using the version of Dymola currently in use at INL, require a license to run. Since running such a model under RAVEN can encompass thousands of individual model runs, they would either have to run sequentially, or more licenses would have to be purchased. There are also two additional licensing options that may be obtained to facilitate running models under RAVEN:[b]

1. *Binary model export* allows executables created using Dymola to be used on other computers without the need for a license.
2. The *Source code generation* feature converts Modelica models into C-language code that may be compiled on any platform.

Both of these options are available at significant additional cost.

For the reasons above, it was decided to investigate the coupling with the OpenModelica implementation of Modelica. There are several implementations of the Modelica modelling language. One such is OpenModelica (https://openmodelica.org/), which is a long-term open-source development effort of the Open Source Modelica Consortium. It has the advantage of being available for the three major platforms of interest: Linux, Mac, Windows. Included in the package is a Modelica language compiler, a visual diagram-based model editing utility, and scripting support. Figure 8 (taken from https://openmodelica.org/openmodelicaworld/tools) illustrates the available tools and how they communicate.

The implementation of the coupling is described in Section 4.2, and Section 4.3 briefly describes the regional application of NHES examined in Garcia et al. 2015 report [3] where the Reverse Osmosis (RO) plant is introduced. Section 4.4 provides the detail of the RO plant and Section 4.5 illustrates its conversion into a model compatible with the OpenModelica implementation of Modelica. Sections 4.6, 4.7, and 4.8 illustrate the process by which RAVEN can be used to run the executable obtained with

[a] https://www.inl.gov/article/new-supercomputer-available-idaho-researchers/.

[b] http://www.3ds.com/products-services/catia/products/dymola/code-and-model-export/.

OpenModelica. This implementation is offered as an example to search the limit surface and to test the capability of two surrogate models to replace the RO plant model.

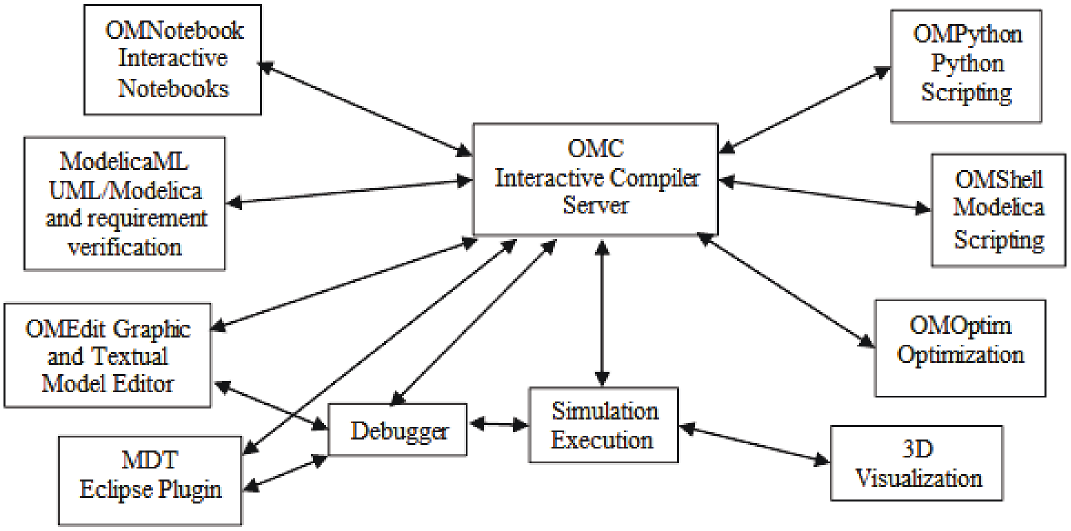

OMNotebook Interactive Notebooks
ModelicaML UML/Modelica and requirement verification
OMEdit Graphic and Textual Model Editor
MDT Eclipse Plugin
OMC Interactive Compiler Server
Debugger
Simulation Execution
OMPython Python Scripting
OMShell Modelica Scripting
OMOptim Optimization
3D Visualization

Figure 8. OpenModelica tools.

4.2 OpenModelica Interface for RAVEN

Since model executables built using OpenModelica are license-free, a specific interface was developed between RAVEN and the executable generated by OpenModelica to start analysis of the coupling between subsystem models and RAVEN. This interface is contained in a Python script file *OpenModelicaInterface.py,* which is already part of the RAVEN code base. The process for creating such an interface is described in Section 17 of the RAVEN User Manual [14]; the specific work is summarized here.

4.2.1 RAVEN OpenModelica Interface

A new code interface is created by writing a Python language file and placing it in the directory in *raven/framework/CodeInterfaces*. Each time RAVEN runs, it automatically searches this folder for code interfaces to load. Each of these interfaces should be created as a Python class that inherits from a base template class *CodeInterfaceBase*. The template imposes the following methods to be coded for each new interface.

The OpenModelica interface for RAVEN implements four of the five code interface functions shown in Table 1, only *checkForOutputFailure()* is not needed. The implementation of the others is now briefly discussed.

Table 1. Code interface methods.

Function	Purpose	Required?
generateCommand(*self*,input,exe,clargs,fargs)	Creates the command that RAVEN will use to invoke the external code.	Yes
createNewInput(*self,inputs,samplerType,Kwargs*)**	Generates a new input file for the external code with parameters changed to RAVEN-specified values.	Yes
finalizeCodeOutput(*self,command,output,workDir*)	Converts output from the external code to Comma-Separated Values (CSV).	No
checkForOutputFailure(*self,output,workDir*)	Provides an alternate way to inform	No

	RAVEN of an external code run failure.	
getInputExtension(*self*)	Informs RAVEN about allowable input file extensions.	No

4.2.2 Function generateCommand()

After using the OpenModelica build environment to convert the Modelica language code to an executable program suitable for the operating system (as discussed in Section 4.6), the code can be invoked as a standalone program using the following syntax:

- `<executable> -f <initialization file> -r <output file>`

  Where:

- `<executable>` is the name of the generated executable program (which will be the same each time the model is run).
- `<initialization file>` is the name of the file containing the initial conditions for the model, which was previously generated using the createNewInput() function.
- `<output file>` is the file that stores the model output.

RAVEN passes to this method the executable name, the initialization file, and the output file, and expects a string returning the command as described above. This structure is necessary so that RAVEN does not own any particular interface, but it is the user that is tasked to implement this method to generate its own specific command. In this way the maintenance of interfaces is not a burden of the RAVEN developers but of the final users, avoiding the issue of proprietary codes and interfaces. The names of the input file and the output file are changed by RAVEN each time the code is run so it is possible to re-construct the input-output mapping when several runs are executed within the same RAVEN run.

4.2.3 Function createNewInput()

This method has the task to produce a new input file for the OpenModelica executable, given the variation of the parameters RAVEN requires to be implemented. RAVEN also passes additional details in the original input file, including the name of the parameters that need to be changed and their new values; in return, RAVEN expects a new modified input file. The names of the parameters to be changed are extended names that contain all the information (syntax) concerning how and where those parameters should be changed in the input file.

OpenModelica model executables read their initial conditions from an eXtensible Markup Language (XML) initialization file (input file) that is created as part of the build process. Since RAVEN already uses XML for many things including input files, the same library ('etree') used for these tasks is employed here. Each time the createNewInput() function is called:

- The original initialization file is loaded into an etree XML object.
- The XML node containing the model initial conditions (called "ScalarVariable") is located.
- Each input space parameter provided by RAVEN is located in this node and changed to its new value.
- The XML object with the modified values is written out to the file name provided by RAVEN in the function call and is returned to RAVEN.

The interface built so far seems capable of accommodating any needed variation of the OpenModelica input file for the NHES studies.

4.2.4 Function finalizeCodeOutput()

The finalizeCodeOutput() function is employed when output files from an external code requires adjustment before being read by RAVEN. Collecting output information can be accomplished in two different ways. First, it could be implemented within the finalizeCodeOuput() method, where a reader imports the data from the code-specific format into the internal RAVEN data. Alternately, Comma-Separated Values (CSV) file can be passed to RAVEN, which then reads it into the internal data. Since OpenModelica executables can be set to produce a CSV file, the second option was the natural choice. An example of a simple OpenModelica CSV output file is shown in the following:

```
"time","h","v","der(h)","der(v)","v_new","foo","flying","impact"
0,3.24834864453,0,0,-9.810000000000001,0,2,1,0
0.002,3.248329024529192,-0.01962000000000001,-0.01962000000000001,-9.810000000000001,0,2,1,0
0.004,3.248270164529105,-0.03924,-0.03924,-9.810000000000001,0,2,1,0
0.006,3.248172064529055,-0.05885999999999999,-0.05885999999999999,-9.810000000000001,0,2,1,0
0.008,3.248034724529015,-0.07847999999999999,-0.07847999999999999,-9.810000000000001,0,2,1,0
```

There are two small compatibility issues that have been addressed in the finalizeCodeOutput() function for OpenModelica. The first issue is that the variable names in the first line of the CSV file, as generated by an OpenModelica model executable, are enclosed in double-quotes (“”), which RAVEN cannot read properly. The other is that lines in some output files were observed to end with commas, which RAVEN interprets to mean that there should be another value present. This function takes care of these problems by reading in a raw CSV output file one line at a time, removing any enclosing double-quotes and trailing commas from each and then writing the modified lines out to a new file. RAVEN then reads the model results from this new file, which contains the original result data in an acceptable form, into his internal formats.

4.2.5 Function checkInputExtension()

Since the input file will always be an XML file, this simple function informs RAVEN that any input file is expected to have a file extension that is common to this file type (any of .xml, .XML, .Xml). This simple function is used for error management, to provide to the users feedback on possible problems encountered while running the code.

4.3 Modeling and Simulation of Regional NHES Configurations in Modelica

In prioritizing region-specific NHES options, available resources, traditional industrial processes, energy delivery infrastructure, and energy users within the selected regions need to be researched to identify attractive clean energy options. INL researchers have identified two region-specific cases for preliminary technical and economic analysis (see Table 2)[1-3].

Table 2. Potential NHES configurations.

Location	Energy and Carbon Source	Energy Product(s)	Variable Energy Load	Integrated Industrial Process
West Texas	Nuclear, wind, natural gas	Electricity, gasoline	Thermal	Natural Gas-To-Liquid (NGTL) fuel
Arizona	Nuclear, solar photovoltaic (PV)	Electricity, fresh water	Electrical	Desalination (via reverse osmosis)

The proposed NHES options were dynamically modeled and implemented in Modelica/Dymola, as shown in Figure 9 and Figure 10 for West Texas and Arizona, respectively. As illustrated in these figures, six main subsystems (components) are modeled for each regional case:

1. Nuclear reactor (water-cooled small modular reactor)

2. Rankine power generation cycle
3. Renewable generation (via wind or solar PV)
4. Power-smoothing battery
5. Electric grid
6. Industrial scale plant (NGTL plant or reverse osmosis desalination plant).

The level of modeling detail within each subsystem varies from mapping functions (e.g., empirical models to estimate renewable energy generation) to more detailed models (e.g., first principle models to predict reverse osmosis performance). In particular, detailed governing equations and models of the NGTL process were initially developed and analyzed using “ASPEN Plus;” simplified formulations were then implemented in Modelica. For additional details on the integrated model, equipment layout, operation, and control for the NHES configurations considered here, see Garcia et al. 2015 [3].

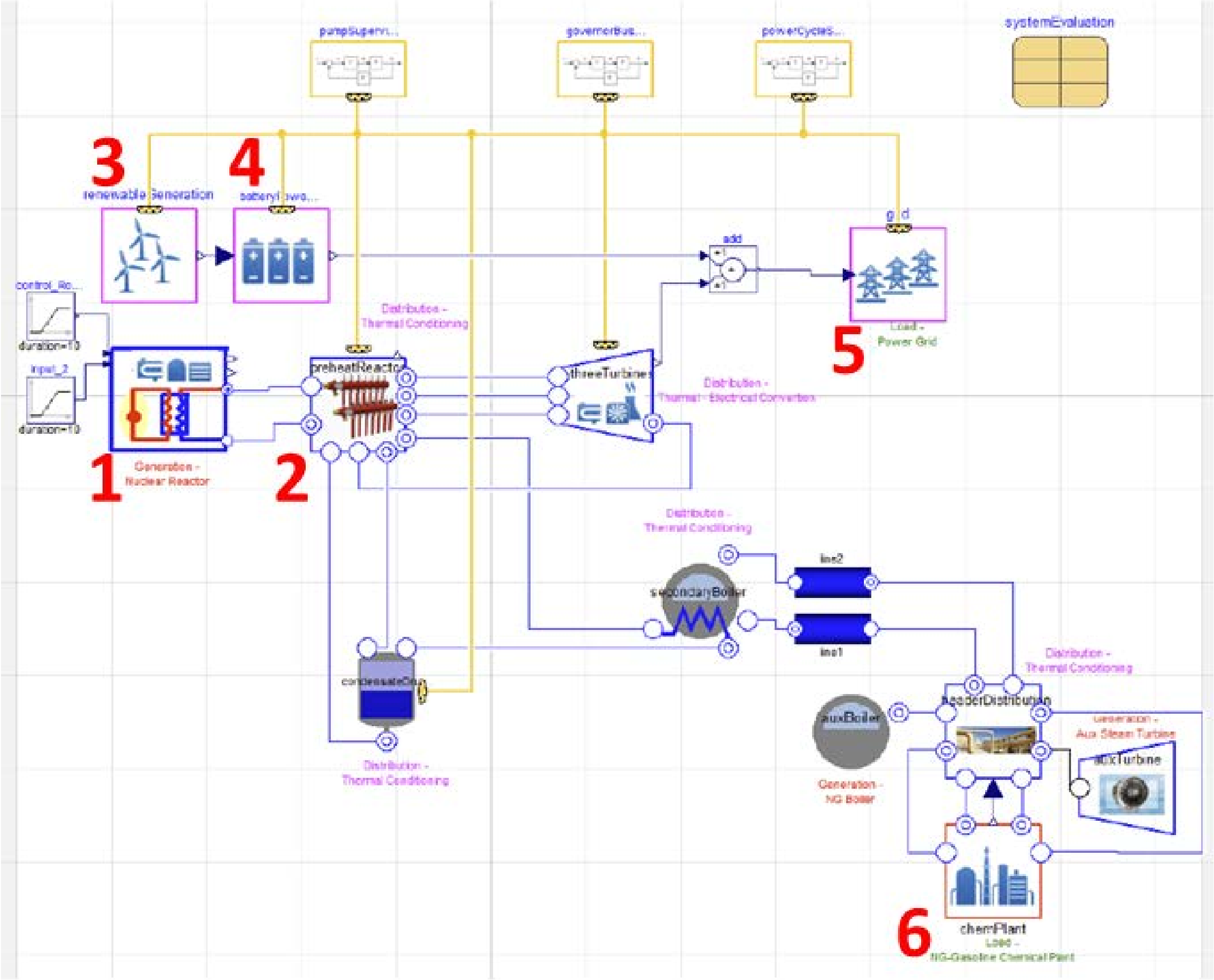


Figure 9. NHES option in West Texas implemented in Modelica/Dymola.

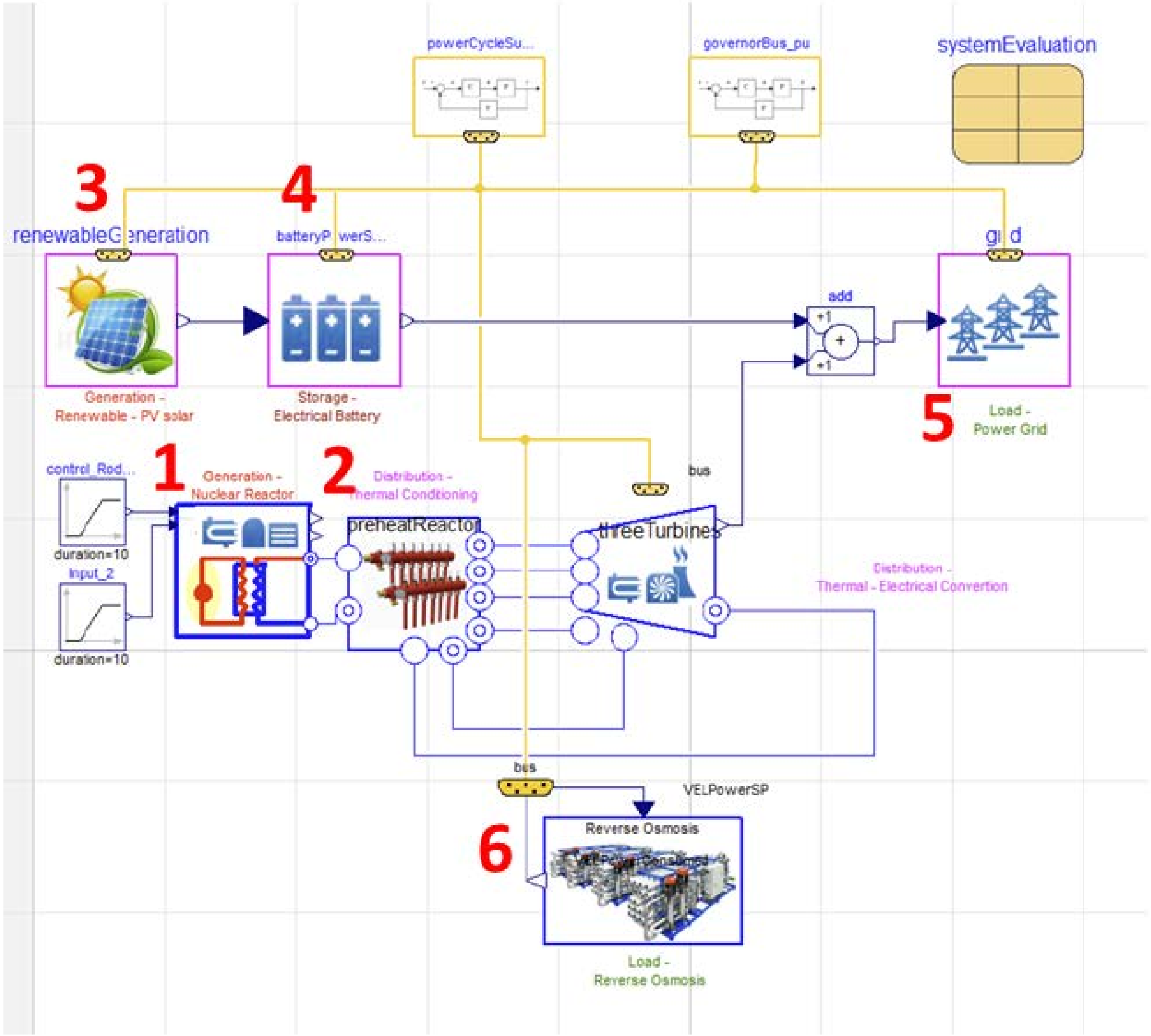


Figure 10. NHES option in Arizona implemented in Modelica/Dymola.

4.4 Reverse Osmosis Desalination

Desalination is a general term for the process of removing salt from water to produce fresh water (permeate). Desalination technologies can be broadly categorized as thermal (phase change) and membrane (non-phase change) processes. Within those two types, there are subcategories depending on different techniques, including multi-stage flash, multi-effect, and vapor compression distillations, electro-dialysis, and reverse osmosis (RO). Of the various methods used for desalination, RO is the predominant means of producing fresh water throughout the world.

The RO process utilizes a semi-permeable membrane, which allows water to pass through, but not salts, to separate the fresh water from the saline feed water. As illustrated in Figure 11(a), a typical brackish water reverse osmosis (BWRO) plant consists of four main components: feed water pre-treatment, high-pressure (HP) pumping, membrane separation, and permeate post-treatment. In this report, the modeling efforts were focused on the two components enclosed in the dashed box shown in Figure 11(a) (i.e., HP pumping and membrane separation). Figure 11(b) depicts the configuration of an RO vessel used in BWRO, which consists of six membrane models connected in series. These pressurized vessels are arranged in rows in each membrane stage, with a two-stage membrane separation being typical. Each stage has a recovery of about 47%, achieving an overall system recovery of 71.6%. When integrated within an NHES configuration, as in the region-specific case studies conducted by INL researchers (Garcia et al., 2015 [3] and Kim et al. 2015 [15]), such a system could manage the high variability of renewable generation and/or electricity demand (load) by utilizing an excess generation capacity, thus acting as a flexible load resource.

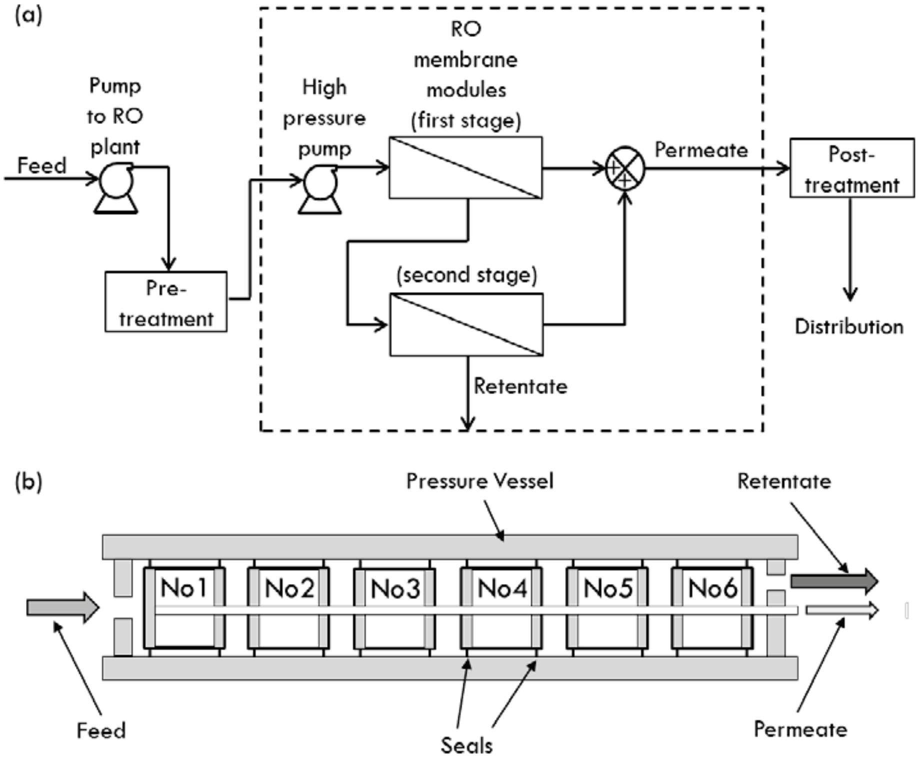


Figure 11. RO desalination: (a) process flow diagram for a two-stage BWRO plant and (b) a schematic of an RO vessel, which consists of six membrane modules in series [2,3].

The BWRO desalination plant was sized for 56,380 m^3/hr (357 million gallons per day [MGPD]) capacity. At this capacity, the plant consumes 45 MW of electrical power to generate the required operating pressure (17.5 bar) for desalting the brackish water, containing 3,502 parts per million (ppm) of total dissolved solids (TDS). The assumed values for the simulation parameters (at nominal operating conditions) used in this report are listed in Table 3. The FilmTech BW30-400 membrane, a spiral-wound membrane manufactured by Dow Chemical, is chosen for simulation. See Garcia et al. 2015 [3] and Kim, et al. [15] for a detailed model of a RO desalination plant and the corresponding control design.

4.5 Conversion of the RO Desalination Plant Model to OpenModelica

As an open-source product, OpenModelica's support for the evolving Modelica language, including equations, algorithms, event-handling, functions, and packages (libraries), lags that of the commercial product Dymola. Therefore, models created in Dymola may contain newer Modelica language features requiring conversion by hand to run under OpenModelica. More importantly, many models created for NHES research utilize libraries created by third parties. Any such libraries used in an OpenModelica model may also need to be converted. The equation solvers provided with Dymola may also be superior to those provided with OpenModelica, resulting in faster simulation time on a Dymola platform. In comparison to OpenModelica, Dymola also supports more robust and reliable initialization of differential-algebraic equations even when the initialization problem turns out to be ill-posed. Despite these disadvantages, it was decided to convert the Dymola model of a BWRO desalination plant for use in OpenModelica, since the model executable built in OpenModelica platform is license-free, hence avoiding dependence on a commercial product at this development stage and allowing the integration with RAVEN to be tested before investing in additional Dymola licenses.

The BWRO desalination system was converted to OpenModelica Connection Editor (OMEdit) by sequentially adding components one at a time. In this manner, it was relatively easy to identify and resolve errors that occurred as a result of incorporating each. After the model was successfully compiled by the OpenModelica complier, additional efforts were needed to help the initialization of the highly nonlinear systems of equations by providing good guess values for a large number of residue states (the values of all other states depend on these states). Finally, the executable model of the RO desalination system was created using the version of OpenModelica.

The component models for this system as they appear in the OMEdit are shown in Figure 12. Key input and output parameters are summarized in Table 3 and Table 4.

Table 3. RO desalination model inputs.

Parameter	Nominal Value	Bounds
Feed pressure [Pa]	1.01×10^5	1.01×10^5 to 1.52×10^5
Feed temperature [°C]	24.85	10 to 26
Feed salinity [ppm]	3,502	3,502 to 10,000
Membrane fouling index [%]	5	0 to 30
Power set point [MW]	45	25 to 45

Table 4. RO desalination model outputs.

Parameter	Nominal Value	Comment
Freshwater production [kg/s] (GPD)	15,588 (3.57×10^8)	None
Average freshwater salinity [ppm]	59.2	A drinking water taste threshold set by U.S. Environmental Protection Agency is 500 ppm.
Water recovery [%]	71.6	It quantifies the fraction of influent water recovered. The water recovery is typically 75% in brackish water RO desalination.
Salt rejection [%]	98.8	It is a characteristic often used by RO membrane manufactures to describe membrane rejection properties. Typically, such membranes achieve NaCl rejections of 98–99.8%.

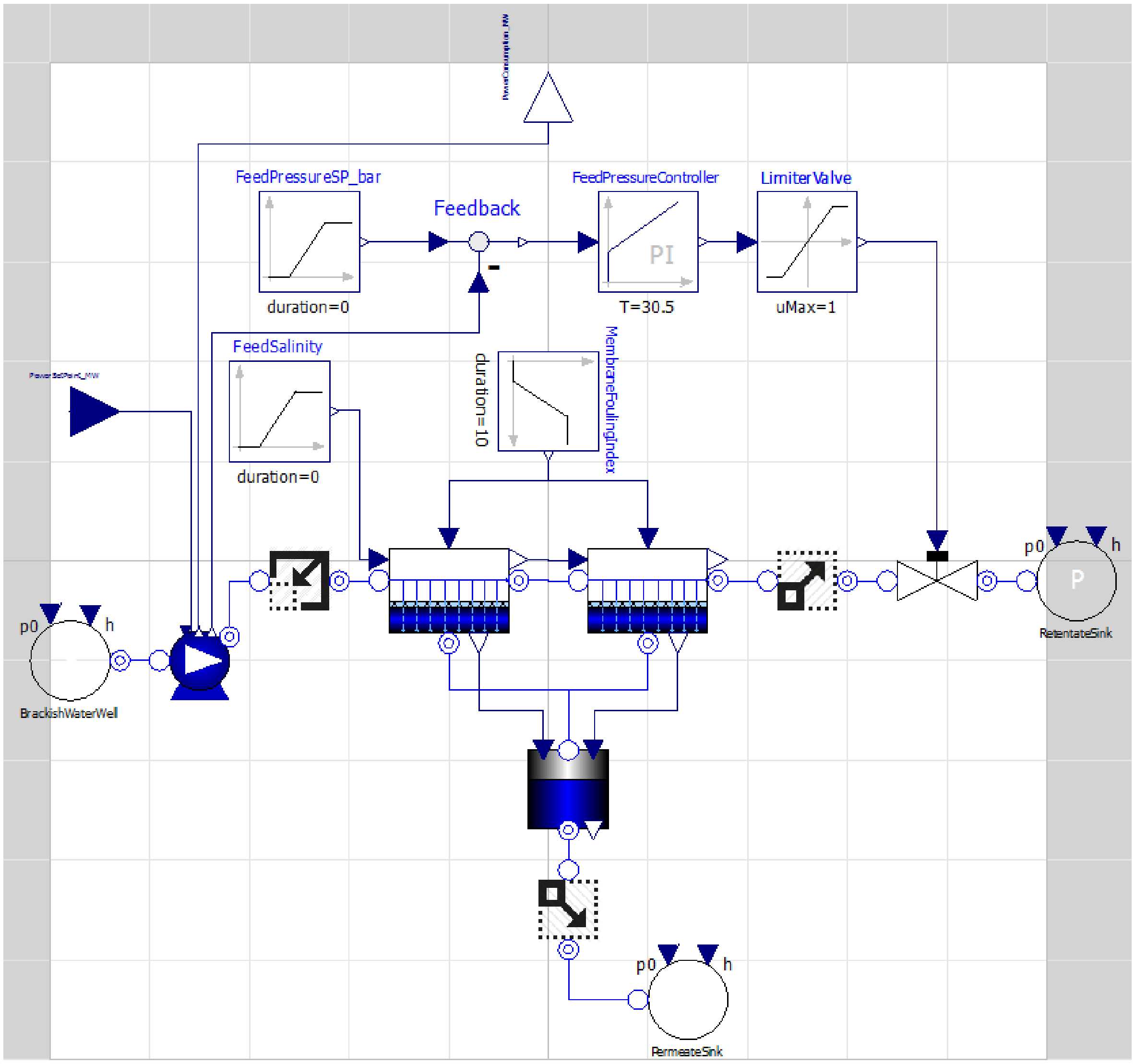


Figure 12. RO desalination plant model in Modelica/OpenModelica.

This model also requires the third-party Modelica-language library "ThermoPower," which is an open-source library for thermal power plant simulation. Table 5 lists the Modelica source files used to build the executable model.

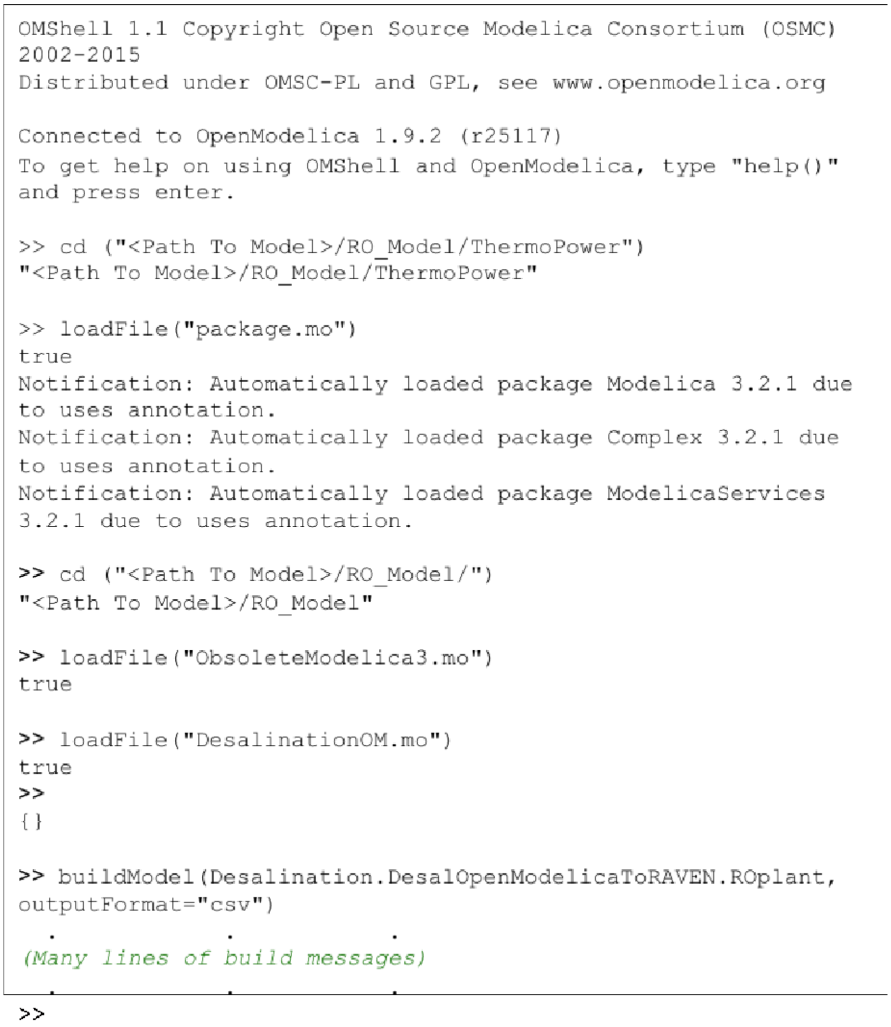
Table 5. Modelica files used in an RO desalination plant.

Name	Number of Code Lines	Purpose
DesalinationOM.mo	1301	RO desalination plant model
ThermoPower (9 Files)	44227	Thermal power plant library
ObsoleteModelica3	3475	Library that contains components from Modelica Standard Library 2.2.2 that have been removed from Version 3.0

4.6 Running the Model with RAVEN

4.6.1 Generation of the OpenModelica Executable Files

Before a simulation can be run, the Modelica code files must be converted into an executable program suitable for the operating system on which it will be executed. The OpenModelica Shell (OMShell) provides one way to do this. After running OMShell on the host operating system, the commands listed in Figure 13 are entered in turn to load the Modelica files into memory, which then enables creation of the executable program.



```
OMShell 1.1 Copyright Open Source Modelica Consortium (OSMC)
2002-2015
Distributed under OMSC-PL and GPL, see www.openmodelica.org

Connected to OpenModelica 1.9.2 (r25117)
To get help on using OMShell and OpenModelica, type "help()"
and press enter.

>> cd ("<Path To Model>/RO_Model/ThermoPower")
"<Path To Model>/RO_Model/ThermoPower"

>> loadFile("package.mo")
true
Notification: Automatically loaded package Modelica 3.2.1 due
to uses annotation.
Notification: Automatically loaded package Complex 3.2.1 due
to uses annotation.
Notification: Automatically loaded package ModelicaServices
3.2.1 due to uses annotation.

>> cd ("<Path To Model>/RO_Model/")
"<Path To Model>/RO_Model"

>> loadFile("ObsoleteModelica3.mo")
true

>> loadFile("DesalinationOM.mo")
true
>>
{}

>> buildModel(Desalination.DesalOpenModelicaToRAVEN.ROplant,
outputFormat="csv")
   .             .             .
(Many lines of build messages)
   .             .             .
>>
```

Figure 13. Command for the creation of the OpenModelica executable.

The *outputFormat="csv"* in the buildModel command set up the executable to produce the CSV output required by RAVEN. After the buildModel command is complete the directory containing the Modelica files will now contain various files from the build process. The needed files are:

- *Desalination.DesalOpenModelicaToRAVEN.ROplant* (on Windows this will be *Desalination.DesalOpenModelicaToRAVEN.ROplant.EXE*). This is the executable file that will perform the model calculations.

- *Desalination.DesalOpenModelicaToRAVEN.ROplant_init.xml*. This file contains the initial conditions for the model. RAVEN's OpenModelica interface will create copies of this file with modified values to vary parameters.
- *Desalination.DesalOpenModelicaToRAVEN.ROplant_info.json*. This file contains transformation steps between different versions of equations within the model, sometimes used for model debugging. The executable will not run unless it can find this file.

4.6.2 Simple RAVEN Input Structure

The files used in an OpenModelica model run are specified in the RAVEN input file. First, the executable and initialization file are specified in the `<Files>` section under `<Simulation>`, as shown below.

```
<Files>
    <Input name="File_Init">Desalination_DesalOpenModelicaToRAVEN_ROplant_init.xml</Input>
    <Input name="File_Exec">Desalination_DesalOpenModelicaToRAVEN_ROplant</Input>
</Files>
```

The OpenModelica executable must also be described in the `<Models>` section under `<Simulation>` in a `<Code>` block (see below). In this case, the name "RO_Plant" is the input that identifies the executable for RAVEN when the execution steps are defined.

```
<Models>
    <Code name="RO_Plant" subType = "OpenModelica">
        <executable>Desalination_DesalOpenModelicaToRAVEN_ROplant</executable>
    </Code>
</Models>
```

Finally, this model may now be referenced in `<MultiRun>` blocks in the `<Steps>` section. In the example considered here, the OpenModelica model "RO_Plant" will be run with inputs derived using the Monte Carlo sampler "MC_FP," with its output being stored in a data structure (PointSet) "RO_PointSet" that can be used in other steps, as well as being printed to the "RO_dump" output object.

```
<Steps>
    <MultiRun name="sample"  re-seeding="200286">
        <Input   class="Files"         type=""              >File_Init</Input>
        <Sampler class="Samplers"         type="MonteCarlo"    >MC_FP</Sampler>
        <Model   class="Models"           type="Code"          >RO_Plant</Model>
        <Output  class='DataObjects'      type='PointSet'      >RO_PointSet</Output>
        <Output  class='OutStreamManager' type='Print'         >RO_dump</Output>
    </MultiRun>
```

4.7 Limit Surface Search

One of the necessary capabilities for the driver of a robust optimization is to identify the location of the limit surface. Within in the algorithm proposed at the beginning of this report the identification of the limit surface is implicit, the approach suggested by Huang et al. [8] would require an explicit location of the limit surface. Since RAVEN already possesses an explicit algorithm for finding the limit surface algorithm, and it is one of the most demanding applications of the code, this application was tested on the INL cluster. The current schema for the search of the limit surface is reported in Figure 14.

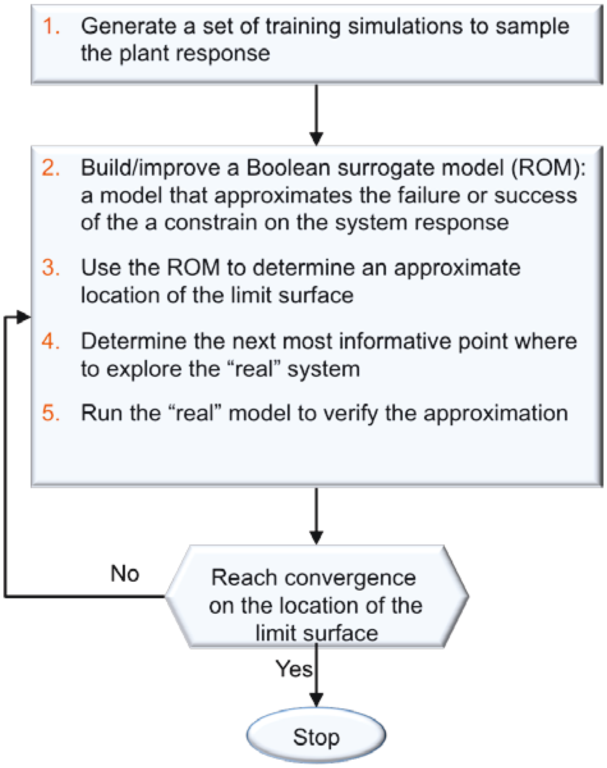

1. Generate a set of training simulations to sample the plant response
2. Build/improve a Boolean surrogate model (ROM): a model that approximates the failure or success of the a constrain on the system response
3. Use the ROM to determine an approximate location of the limit surface
4. Determine the next most informative point where to explore the "real" system
5. Run the "real" model to verify the approximation
No
Reach convergence on the location of the limit surface
Yes
Stop

Figure 14. Limit surface search algorithm.

Note that the search schema used the construction of a surrogate model to accelerate the calculation, similar to the way the envisioned driver, using the approach suggested by Huang et al. [8], would construct the admissible space given the reliability test of the grid.

The problem tested requires:

- Freshwater Production (kg/s) ≥ 2.5e+ 8
- Average Freshwater Salinity (ppm) ≤ 200.

While the varying parameters are:

- Feed temperature (°C)
- Feed salinity (ppm)
- Power set point (MW).

Figure 15 illustrates the space where there is no solution for the given constraints. There is a numerical artifact on top of the figure (set of isolated points) that currently needs more investigation.

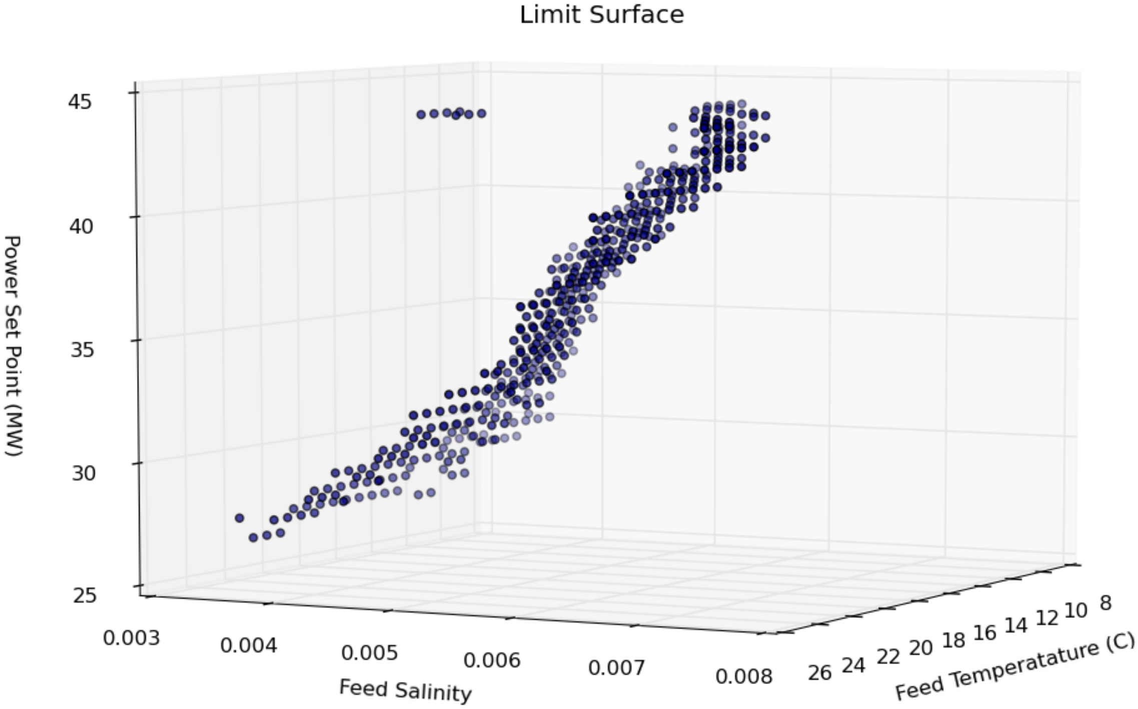


Figure 15. Limit surface.

4.8 Surrogate Model Test

As mentioned, RAVEN is able to rapidly construct and evaluate surrogate models of the NHES that could be crucial in creating a computationally affordable framework for the economic analysis of NHES. To test this capability, the reverse osmosis desalination plant was considered.

The set of parameters that have been changed are:

- Feed pressure (Pa)
- Feed temperature (°C)
- Feed salinity (ppm)
- Membrane fouling index (%)
- Power set point (MW).

The FOM considered are:

- Freshwater production (GPD)
- Freshwater production (kg/s)
- Average freshwater salinity (ppm)
- Water recovery (%)
- Salt rejection (%).

The possible variation ranges of the input parameters are provided in Table 6.

Table 6. Input parameters (nominal and bound value) for the RO plant.

Parameters or Variable	Units	Nominal Value	Lower Bound	Upper Bound
Feed pressure	Pascal	101325	101325	151987.5
Feed temperature	Celsius	24.852	10	26
Feed salinity (concentration)	ppm (part per million) or mg/kg	3502	3502	10000
Membrane fouling index	%	5	0	30
Power set point (or variable electrical load [VEL])	MW	45	25	45

The testing of the capabilities of the surrogate models has been performed using several training sets and validation sets. The mean of the errors and its standard deviation are reported in Table 8 through Table 10, while Table 7 reports the mapping between training sets and validation sets for each test. As may be inferred from the table, two surrogate models have been tested. The linear regressor is a least-squares fitting of the response of the system (figures of merit) for a linear representation (linear multivariate regression). The inverse distance weighted regressor is instead an interpolation scheme where the value of the system response for a given input is computed as the contribution of the system response for the whole training set. Each point of the training set contributes proportionally to the distance between its input coordinate and the input coordinate of the inquired point.

RAVEN possesses more than 20 surrogate models. The linear regressor was chosen for its simplicity, and the inverse weight model was selected for its robustness and versatility in interpolating even a complex response function.

Table 7. Data set descriptions used for creation of the surrogate models and their cross -validation.

Training Set	Verification Set	
	Monte Carlo 1	Monte Carlo 2
Monte Carlo 1: 2500 random sampling using uniform distributions	Table 8	Table 9
Monte Carlo 2: 2500 random sampling using uniform distributions	The Monte Carlo set 2 was used only for verification not for training	
Grid 1: equally spaced 3125 point in Cartesian grid (5 point in each dimension)	—	Table 10

The repeatability of the exact sequence of samples is ensured by controlling the seeding of the pseudo random number generator in RAVEN. Notice that, overall, the inverse weight performs better (as expected) in simulating the overall behavior of the system. This analysis was performed on the INL cluster Falcon with eight cores, using the parallel management infrastructure of RAVEN and the installation of OpenModelica on the cluster.

Table 8. Comparison of ROMs versus original model (training set: Monte Carlo 1, cross-validation set Monte Carlo 1).

Linear Regressor ROM					
FOM	**Freshwater Production (GDP)**	**Freshwater Production (kg/s)**	**Average Freshwater Salinity (ppm)**	**Water Recovery (%)**	**Salt Rejection (%)**
Mean error (%)	-0.03%	-0.28%	-0.13%	0.99%	0.20%
Error Standard deviation	0.001748556	0.043634653	0.049000767	0.087837841	0.034317013
Inverse Weight ROM					
FOM	**Freshwater Production (GDP)**	**Freshwater Production (kg/s)**	**Average Freshwater Salinity (ppm)**	**Water Recovery (%)**	**Salt Rejection (%)**
Mean error (%)	0.00%	0.00%	0.00%	0.00%	0.00%
Error Standard deviation	0	0	0	0	0

Table 9. Comparison of ROMs vs. original model (training set: Monte Carlo 1, cross validation set Monte Carlo 2).

Linear Regressor					
FOM	**Freshwater Production (GDP)**	**Freshwater Production (kg/s)**	**Average Freshwater Salinity (ppm)**	**Water Recovery (%)**	**Salt Rejection (%)**
Mean error (%)	-0.03%	-0.04%	0.05%	1.22%	0.31%
Error Standard deviation	0.002840228	0.071352009	0.072836712	0.150156766	0.052742815
Inverse Weight ROM					
Mean error (%)	0.02%	0.28%	0.28%	1.23%	0.04%
Error Standard deviation	0.001537467	0.037357812	0.037298828	0.078756227	0.027160556

Table 10. Comparison of ROMs vs. original model (training set: Grid, cross validation set Monte Carlo 2).

Linear Regressor					
FOM	**Freshwater Production (GDP)**	**Freshwater Production (kg/s)**	**Average Freshwater Salinity (ppm)**	**Water Recovery (%)**	**Salt Rejection (%)**
Mean error (%)	0.01%	5.49%	0.44%	-1.93%	-1.32%
Error Standard deviation	0.002975828	0.146102759	0.096586702	0.216198605	0.073214743
Inverse Weight ROM					
FOM	**Freshwater Production (GDP)**	**Freshwater Production (kg/s)**	**Average Freshwater Salinity (ppm)**	**Water Recovery (%)**	**Salt Rejection (%)**
Mean error (%)	-0.03%	0.14%	0.14%	1.47%	0.25%
Error Standard deviation	0.000651345	0.016272846	0.016247131	0.033980499	0.01348851

5. DEMONSTRATION OF THE ECONOMIC MODEL

The following section presents a simplified economic model that was developed to initiate an approach to analyzing the dynamics of economic performance.

5.1 Model Description

One of the needs of the NHES research program is to understand the value proposition for NHES to the electric grid as a whole. To do so, the FOMs necessary to characterize the NHES value proposition must be determined, and an economic analysis tool or suite of tools must be used or developed to provide accurate estimates of those FOMs. To proceed with these two tasks, a preliminary economic analysis tool was developed using RAVEN, which was originally developed to conduct probabilistic safety analysis by wrapping around and driving specialized nuclear safety codes. RAVEN has been further developed to drive any supplied internal or external code. In the current instance, RAVEN seeks to clarify the operational space for a simplified Renewable Energy (RE)/Natural Gas (NG)/NHES system. Though it may eventually provide a unique analysis capability for the NHES research program as a part of the larger suite of electric grid analysis tools maintained by DOE, the near-term objective of this tool is to help clarify the needs of future analysis efforts, which will be undertaken at a much larger and more complex scale.

In the current version, the preliminary economic analysis tool exists as an “external model” within RAVEN. External models are codes written in Python that are directly interfaced with RAVEN via an Application Programming Interface (API). To aid in understanding the structure of the preliminary economic analysis tool, a flowchart is provided in Figure 16. RAVEN acts as the driver for the model, where it iterates the model parameters, then stores and visualizes the results.

For each energy source the following data are kept constant and stored inside the model:

- D: discount rate
- N: lifetime of the plant in years
- CRF: credit recovery factor computed as $CRF = \frac{D*(1+D)^N}{((1+D)^N-1)}$

- Capital Cost
- T: tax rate
- D_{PV}: present value of depreciation
- $fixed\ O\&M$: fixed operation and management costs
- $variable\ O\&M$: variable operation and management costs
- Fuel price
- Heat factor.

The electricity demand is also stored inside the model and is taken from the ERCOT database of hourly demand. The parameters perturbed by RAVEN, which constitute the input space, include the following:

- Fraction of effective renewable capacity plus hybrid capacity installed. The effective installed capacity is computed as the installed capacity of renewable scaled by the credit capacity factor.
- The ratio of the effective renewable capacity versus the hybrid capacity.
- The credit capacity factor of the renewable.

It should be noted that the credit capacity factor used by the model corresponds to the fraction of the renewable resource installed capacity that is estimated to be available to cover the actual peak demand. With these parameters and data the code constructs the total installed capacity such that:

(Capacity Renewable)*Credit Capacity Factor + Capacity Wind + Capacity Hybrid = max{hourly demand}

For each value of the above parameters, each time RAVEN runs the model, the model computes the hourly availability of renewable-supplied electricity using a gamma (alpha=2, gamma=5) distribution. The electricity supplied by the renewable is subtracted from the demand, resulting in the net demand that is first filled by the natural gas capacity and then, if more is needed, by the hybrid systems.

The dispatching is organized according to the following assumption:

- The renewable is the source having the lower marginal cost and, therefore, is the first to be in the stack.
- The hybrid system, when not selling electricity, sells a co-product that has a marginal cost above the one incurred by natural gas producing electricity.

Once the dispatching is known for the whole time covered by the demand data, it is possible to compute the effective utilization of the plants and to compute the Levelized Cost of Electricity (LCOE) by the formula:

$$LCOE = 10 * \left(\frac{Capital\ Cost * CRF * (1 - T * D_{PV})}{8760 * Capacity\ Factor * (1 - T)} + \frac{fixed\ O\&M}{8760 * Capacity\ Factor} + \frac{variable\ O\&M}{1000} + \frac{Fuel\ Price * Heat\ Rate}{1000000}\right)$$

Note that the "Capacity Factor" is not driven by the availability of the plant, but is replaced by the effective utilization of the plant. The LCOE computed is the cost of electricity that would lead to a NPV of 0 (zero) given the actual utilization of the plant. For the hybrid system we have assumed that the capacity factor is 100% since the system always has the option to sell the co-product. It is also assumed that the hybrid plant switches between selling electricity and selling the co-product when the prices are the same. Under this assumption the impact on the profit from selling the co-product, thereby reducing the LCOE, is accounted by the 100% capacity factor.

An important feature of the model is that it allows for the possibility that the demand is not always met. The system is oversized in terms of installed capacity since only a fraction of the renewable installed capacity is accounted in covering the maximum demand. This does not imply that the demand is always met, since the gamma distribution used to model the wind distribution could lead to a wind capacity available in a specific moment in time that is below the value of the credit capacity factor. If this scenario occurs at the same time as the peak demand, it could lead to a lower available capacity than what is needed. When such a situation arises, the model imports electricity at a fixed high cost from outside the considered system and detects the failure to meet demand.

Finally, the model output reports the cost of electricity that was needed to ensure coverage of demand, the number of hours the system was incapable of meeting demand, CO_2 emissions, LCOE for each source, and the utilization fraction for each source. The CO_2 emissions from the imported electricity have been set equal to the amount produced by natural gas plants. Figure 16 provides the flow chart for the simulation.

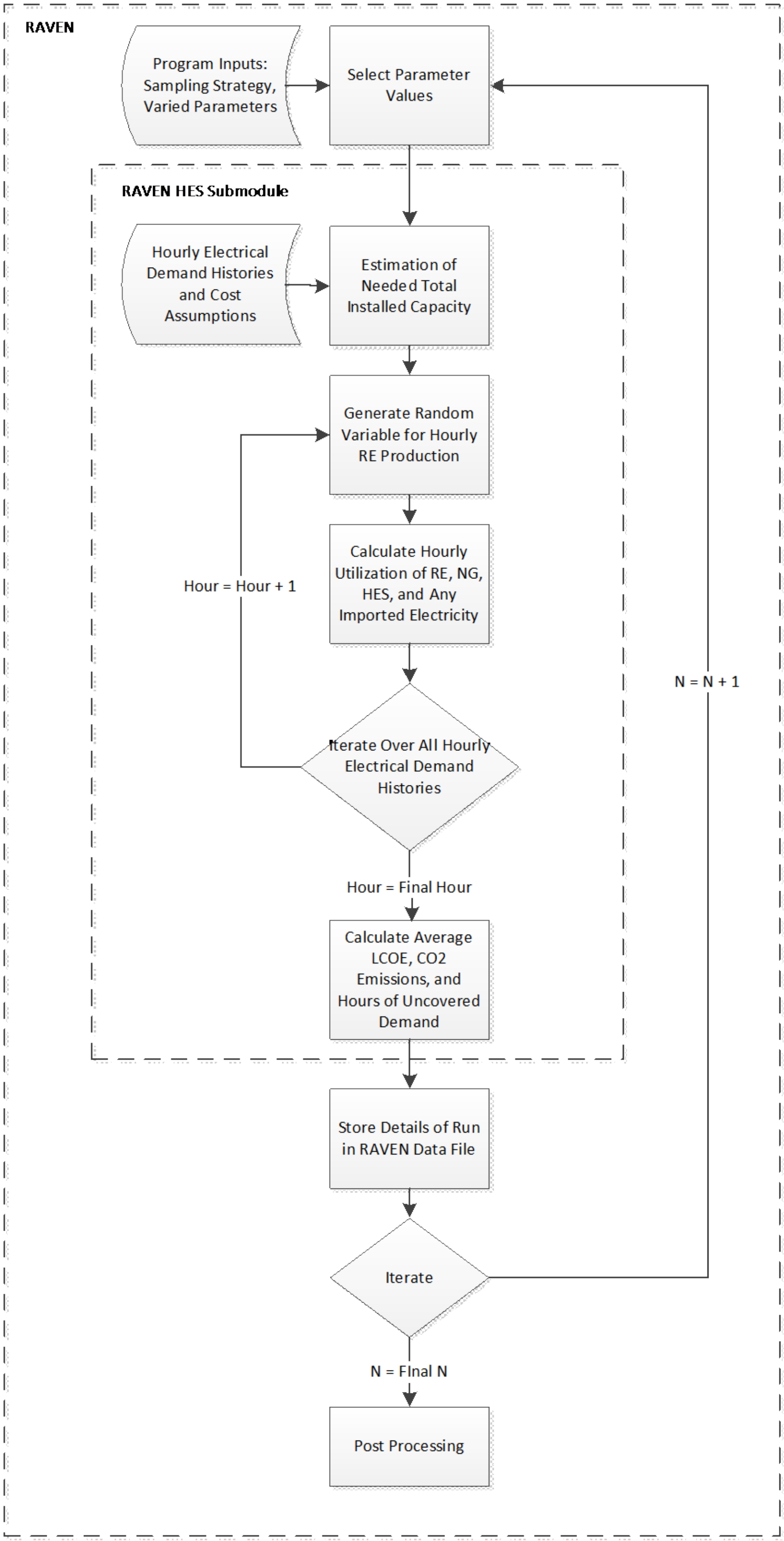

RAVEN
Program Inputs: Sampling Strategy, Varied Parameters
Select Parameter Values
RAVEN HES Submodule
Hourly Electrical Demand Histories and Cost Assumptions
Estimation of Needed Total Installed Capacity
Generate Random Variable for Hourly RE Production
Hour = Hour + 1
Calculate Hourly Utilization of RE, NG, HES, and Any Imported Electricity
Iterate Over All Hourly Electrical Demand Histories
N = N + 1
Hour = Final Hour
Calculate Average LCOE, CO2 Emissions, and Hours of Uncovered Demand
Store Details of Run in RAVEN Data File
Iterate
N = FInal N
Post Processing

Figure 16. Flow chart for the economic assessment demonstration tool.

5.2 Results Discussion

Although a simplified example was conducted, it immediately highlighted important features that are key in justifying the selected approach and supporting the validity of the tools.

5.2.1 Parametric Studies

Figure 17 shows (left side) that there are two relative maximum values in the total LCOE for the modeled system. One maximum value corresponds to the minimum value of the credit capacity factor; the other maximum corresponds to the maximum value of the credit capacity factor. This is explained by the fact that for a very low credit capacity factor, the volatility of the wind is underestimated; therefore, the LCOE is pushed up by the high price of imported electricity required to meet demand. For low values of the credit capacity factor, the effective capacity of the renewables is underestimated and the LCOE suffers from a large capital cost with respect to the required capacity.

This result is confirmed by observing Figure 18, where the color map represents the total fraction of renewable capacity installed (greater than one since it is divided by the credit capacity factor). The plot clearly shows the problem of having oversized the capacity of the renewable in the lower-left corner. The problem with using a high-capacity factor for the renewable is alternately confirmed by the next plot Figure 19, where the number of hours in which there was insufficient capacity to meet demand is plotted. The figure confirms that the number of such events escalates with increased renewable penetration and with increased credit capacity factor.

The last plot provided in Figure 20 is also very informative, as it shows that overestimating the reliability of the wind creates an increase in CO_2 emissions because imported electricity is very costly in terms of CO_2 production. Underestimating the renewable availability will produce a very low level of emissions, but at the same time results in a higher LCOE overall due to the unused capacity. Moreover, the CO_2 emissions clearly show a positive sensitivity to the fraction of natural gas in the mix.

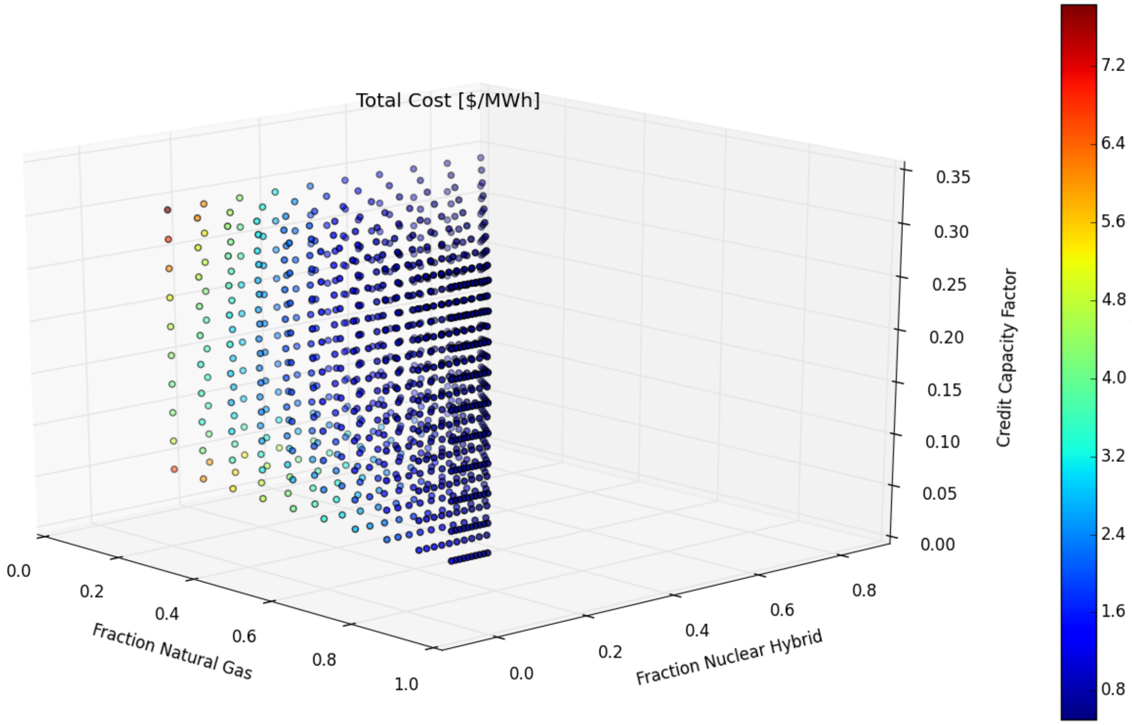


Figure 17. LCOE parametric studies (Total cost is equivalent to LCOE_Total).

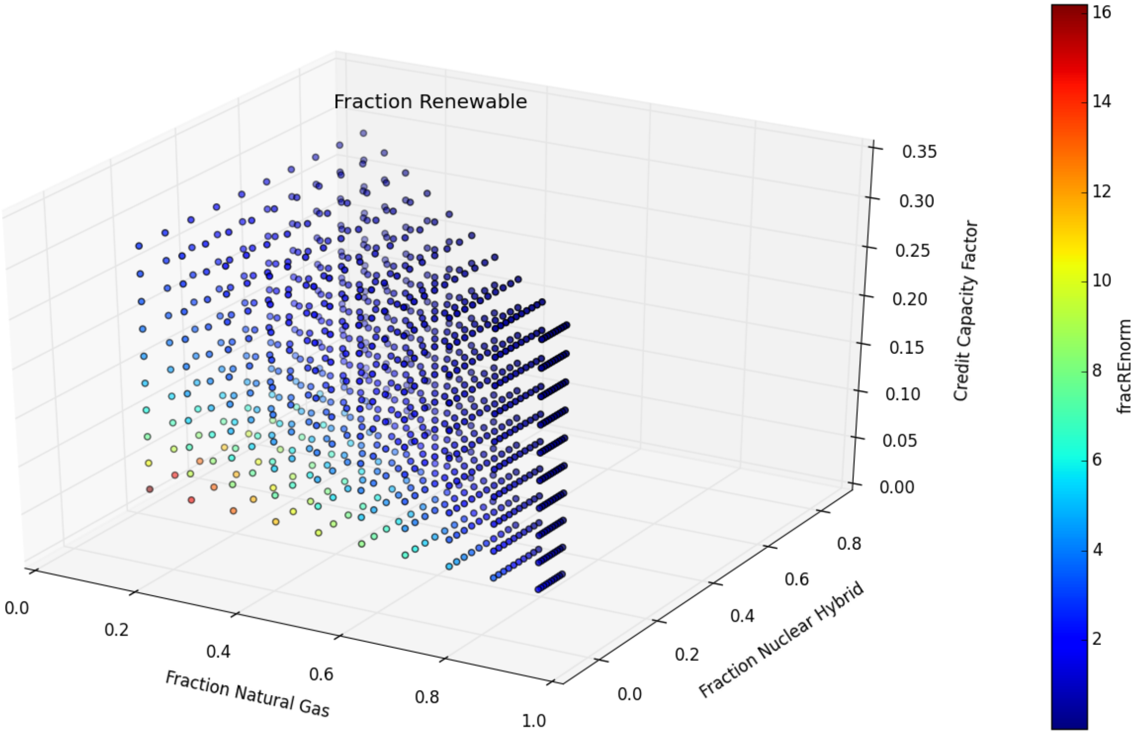


Figure 18. Total renewable capacity installed.

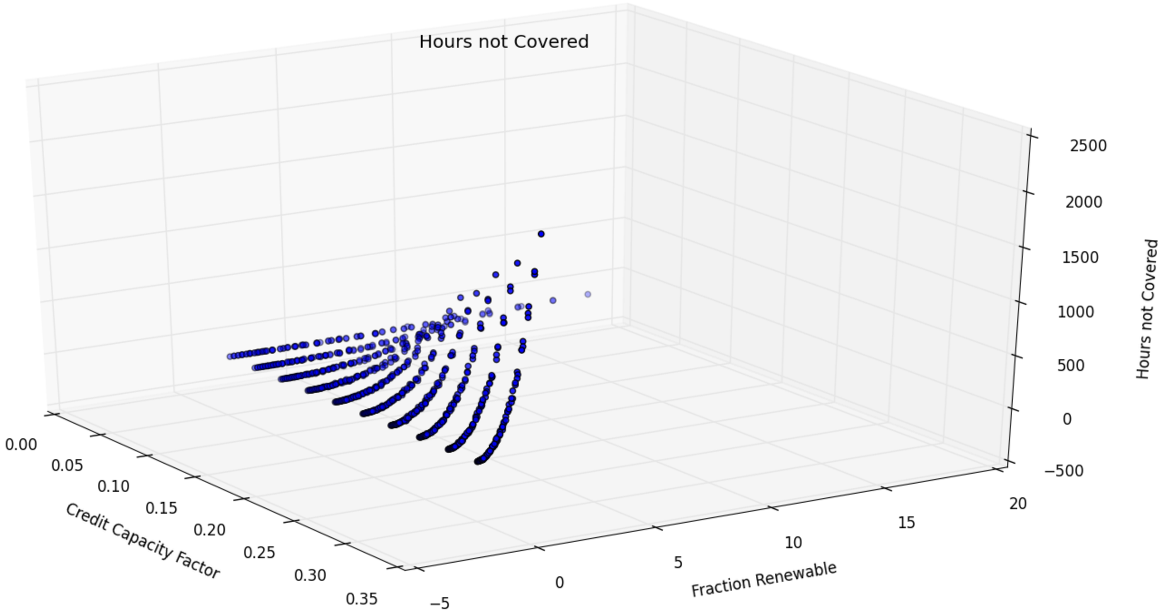


Figure 19. Hours for which demand cannot be met over a one-year period.

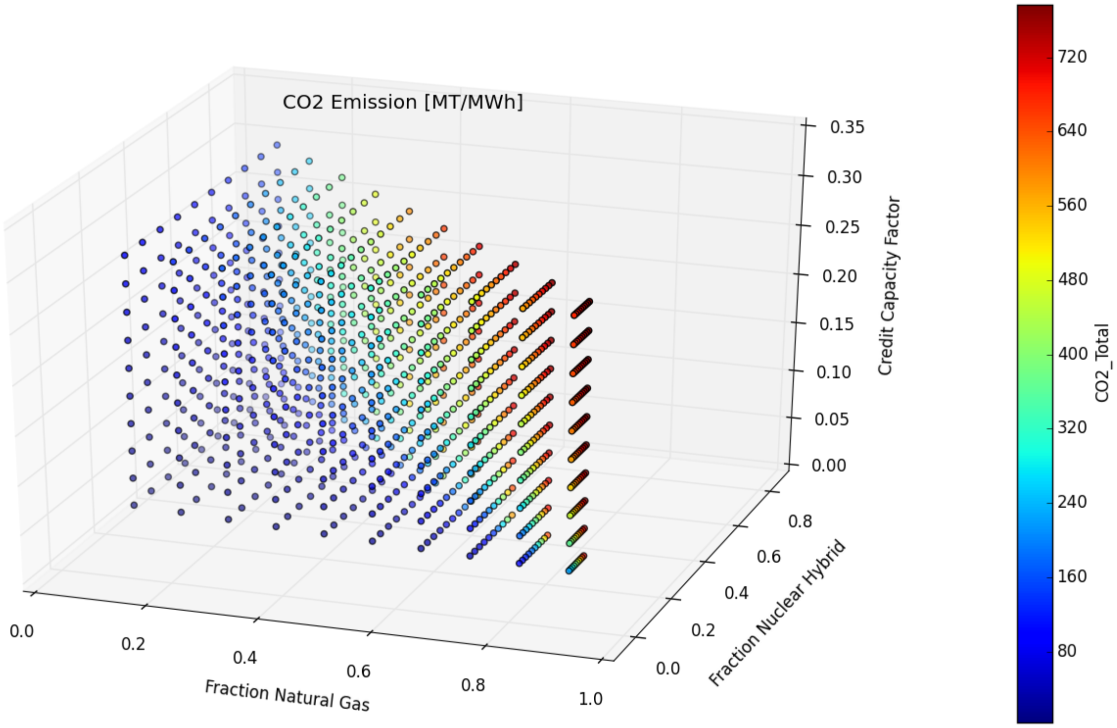


Figure 20. LCOE parametric studies.

5.2.2 Reliability Surface Analysis

The last analysis performed in the current work was a search of the generation source mix that allows the system to not exceed 30 hours for which demand exceeds the dispatchable capability over the defined 1-year time frame. This surface is shown in Figure 21. There are several important aspects of this test. First, the limit surface found is actually a de-noised limit surface. Since the renewable availability is a random value determined inside the model (beta distribution sampling), this impairs the existence of a deterministic relationship between the input space and the output space. In other words, two different simulations with the same input parameters might lead to different results given the probabilistic behavior of the system. In those cases the limit surface is defined in a probabilistic sense; filtering techniques are typically used to make the problem treatable. RAVEN applies a de-noising technique by testing the convergence of the limit surface using the prediction of a surrogate model rather than the real model. The surrogate model acts as a statistical filter. There are, of course, risks defining tolerances on the surrogate models that are too tight, which could lead to very poor convergence and high computational costs.

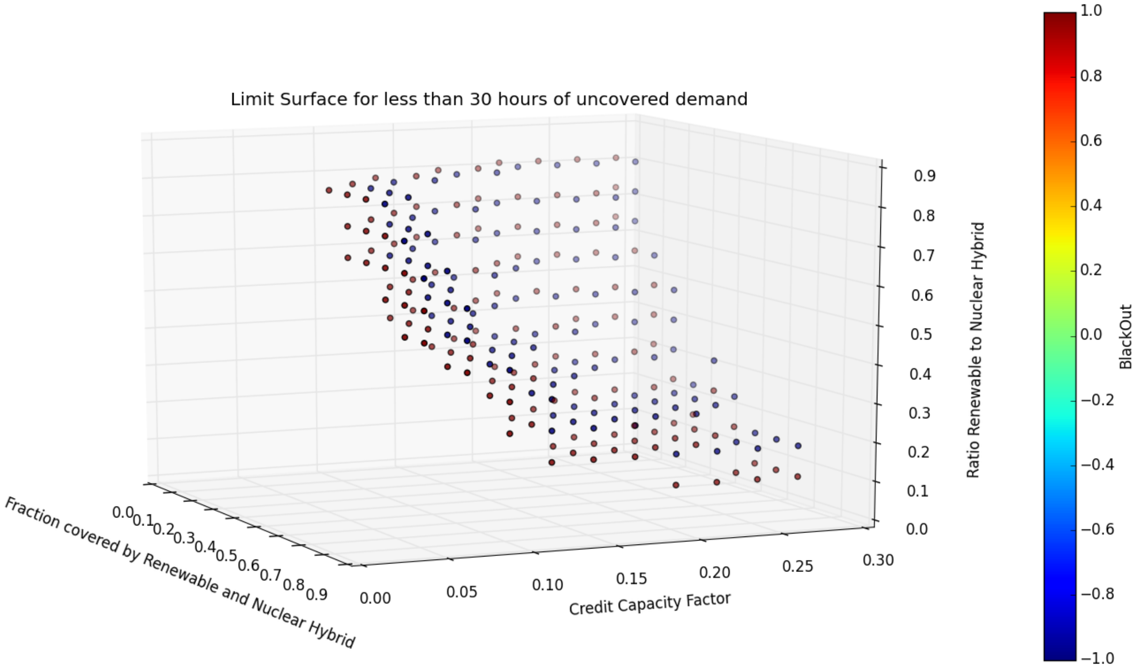


Figure 21. Limit surface illustrating the parametric region where the system will not lead to more than 30 hours of loss of load over a year time frame.

More research is needed to optimize the defined algorithms, but the initial results are encouraging. Figure 21 illustrates the limit surface found by directing the code to perform the search using a grid for which each element is sized to be 1/1000 of the total hypervolume of the input space and using a surrogate model as a “support vector machine” with a radial basis function as a kernel [16].

The red colored dots lay on the side of the limit surface facing the region of the input space in which the 30-hour limit would be exceeded. As expected, the limit surface tends to be tangent to the planes:

- Ratio of the renewable to hybrid equals 0
- Ratio of the renewable plus hybrid capacity to total capacity equals 0
- Credit capacity factor equals 0.

These are all cases in which the variable renewable is removed from the system. As the fraction of the capacity provided by the renewable plus nuclear hybrid increases, the risk of blackout also increases, as does the credit capacity factor (overestimation of renewable availability).

While the result would be more immediately understood if the plot was expressed in the transformed coordinate, as was done for the previous results (e.g., fraction of renewable, fraction hybrid etc.), this is not yet feasible since the limit surface could only be expressed with respect to the original input parameters. In this case, a clustering-based analysis, also available in RAVEN, would be more effective in visualizing the results; however, the point of the current work was to investigate the applicability of the current algorithms to search for the limit surface of the NHES analysis under a stochastic behavior of the system.

6. ENHANCEMENT OF THE DEVELOPMENT FRAMEWORK: RUNNING RAVEN UNDER WINDOWS™ OS

Most RAVEN software development takes place on Mac and Linux operating systems, while several of the modeling tools used for the evaluation of the NHES to date are under Windows. Therefore, it has been decided to have a RAVEN version running under Windows to facilitate the integration of the teams and the various work tasks. One of the key problems in adding a new platform to those that are already supported is the need to extend the regression testing system to the new OS—Windows in this case. RAVEN has a strong quality assurance (QA) pedigree, most of which derives from a complex automated regression test system. The regression system was inherited and extended from the MOOSE development environment that does not, unfortunately, cover Windows.

To facilitate early detection of problems due to software changes under Windows, a new automatic software testing facility was created. A specific small Windows server has been dedicated to performing the tests. A set of scripts was written to perform the following sequence of operations:

1. The current *devel* branches for RAVEN and Crow (the RAVEN statistical library) packages are checked out from the software repository.
2. If Crow has changed since the last time the script was run, try to rebuild it.
3. If the Crow build fails, send an e-mail message to the RAVEN developer's mailing list indicating the problem, and stop.
4. If RAVEN has changed since the last time the script was run, try to rebuild the AMSC module.
5. If the AMSC module build fails, send an e-mail message to the RAVEN developer's mailing list indicating the problem, and stop.
6. If either Crow or RAVEN has changed, run the framework tests and collect the output.
7. Send the output in a message to the RAVEN developer's mailing list. The subject line of this message is the count of tests that pass, fail, and were skipped.

The primary script (written in WindowsPowerShell) is run every night and performs these functions by calling several Unix-style shell scripts. Each time the RAVEN or Crow code is updated in the repository, developers can expect to see the results of the regular test suite as e-mail the next day.

This regression testing task was necessary to ensure the needed software reliability using RAVEN to the modelers using, for example, Dymola (currently available only under Windows) or PSCAD.

7. FUTURE ACTIVITIES

The activities performed in FY15 provide a clearer picture of the next steps required to implement the framework for the economic optimization of NHES designs. The short-term goal in further developing the framework should probably be the integration of the regional case studies under the RAVEN umbrella to integrate the analysis that has been performed to date by introducing a stochastic model of the demand. If this is selected as the next task, then the following tasks need to be completed:

- Modify the RAVEN interface for OpenModelica to be compatible with the Dymola implementation of Modelica
- Acquire an unlimited processor license for Dymola for the Linux environment

- Complete the optimization driver, in particular:
  - Integrate the capability to generate a random representation of the time history of both demand and available renewable
  - Extend the set of optimization algorithms available
  - Ensure that the proper figures of merit are available after each run (u, n, c)
  - Ensure that the optimization parameters are available as part of the input parameters (e.g., capacity of each subsystem)
- Acquire the time history for the regions over a fairly large time window.

The above capabilities will allow one to perform a full-scale evaluation of the feasibility and the value of the economic analysis framework proposed. At the same time, this work could initiate the activity to evaluate Ptolemy II (or other frameworks) as a hub for coordinating the communication of multiple subsystems for more complex cases. Moreover, it is necessary to create the infrastructure to allow cooperation of the various laboratories involved in the NHES research, such as a gitlab repository to ensure version control and quality control by a regression test system (this approach will also leverage the infrastructure built for the development of the MOOSE framework under github). This infrastructure will also allow for the discussion and the resolution of any Intellectual Property issues that will naturally emerge in a large, shared effort.

8. CONCLUSIONS

A mathematical formulation of the problem of cost minimization for NHES integration has led to a better understanding of a software infrastructure and algorithms that are computationally efficient for the economic evaluation of NHES. The main problem can be been cast as a robust constrained optimization problem.

A literature review did not identify any existing algorithms for the solution of the problem, in particular with respect the implementation of stochastic constraints. This report proposes a possible new approach and shows preliminary results of the implementation in RAVEN.

An overall software infrastructure, based on RAVEN and OpenModelica coupling, has been developed and demonstrated. The portability of the software has been tested and enhanced using RAVEN under Windows and OpenModelica in the INL HPC cluster. The coupling between RAVEN and OpenModelica was tested on various platforms and applications. Among these tests, a proof of concept for the representation of a component of the NHES system (desalination plant) by a surrogate model was successfully completed.

Steps were also made toward a portable common development platform, including porting the RAVEN regression test system under Windows, to ensure a proper QA level is maintained under the platform used by the developers of NHES models.

Finally, a simple economic model was developed to test both the capability of RAVEN to identify limit surfaces representing stochastic constraints and to confirm the dependency of the system's economic performance (electricity cost) on the fraction of variable renewable sources (wind, in the specific case used in the initial example). In particular, results presented in this report indicate that the cost of electricity has a minimum for a specific fraction of renewable resources such that their utilization is maximized while the volatility introduced is managed.

Overall, the tasks performed began the implementation of the much-needed capabilities to optimize and evaluate NHES. The results of this effort establish a clear path for future work that will expand, refine, and utilize this capability.

9. REFERENCES

1 Rabiti, C., H. E. Garcia, R. Hovsapian, R. A. Kinoshita, G. L. Mesina, S. M. Bragg-Sitton, R. D. Boardman, *Modeling, Simulation and Control Gap Analysis Report*, INL/EXT-15-34877, April 2015.

2 Elmqvist, H., S. E. Mattsson, "Modelica—The Next Generation Modeling Language an International Design Effort," *Proceedings of the 1st World Congress on System Simulation (WCSS'97), Singapore, September 1–3, 1997*.

3 Garcia H. E., J. Chen, J. S. Kim, M. G. McKellar, W. R. Deason, R. B. Vilim, S. M. Bragg-Sitton, R. D. Boardman, *Nuclear Hybrid Energy Systems—Regional Studies: West Texas & Northeastern Arizona*, INL/EXT-15-34503, April 2015.

4 Blochwitz, T., M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, D. Neumerkel, H. Olsson, J.-V. Peetz, S. Wolf, "The Functional Mockup Interface for Tool Independent Exchange of Simulation Models," *Proceedings of the 8th Modelica Conference, Dresden, Germany, March 2011*.

5 Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, "Scikit-Learn: Machine Learning in Python," *Journal of Machine Learning Research*, Vol. 12, 2011, pp. 2825–2830.

6 Gerber, S., K. Potter, "Data Analysis with the Morse-Smale Complex: The msr Package for R," *Journal of Statistical Software*, Vol. 50, No. 2, July 2012, pp. 1–22.

7 Spall, J. C., "Stochastic Optimization," in *Handbook of Computational Statistics: Concepts and Methods (2nd edition)*, J. Gentle, W. Härdle, and Y. Mori, eds., Springer-Verlag, Heidelberg, Chapter 7, pp. 173–201.

8 Huang, D., T. T. Allen, W. I. Notz, N. Zeng, "Global Optimization of Stochastic Black-Box Systems via Sequential Kriging Meta-Models," *Journal of Global Optimization*, Vol. 34, No. 3, March 2006, pp. 441–466.

9 Spall, J. C., "An Overview of the Simultaneous Perturbation Method for Efficient Optimization," *Johns Hopkins APL Technical Digest*, Vol. 19, No. 4, 1998, pp. 482–492.

10 Spall, J. C., "Adaptive Stochastic Approximation by the Simultaneous Perturbation Method," *IEEE Transactions on Automatic Control*, Vol. 45, No. 10, October 2000, pp. 1839–1853.

11 Flores, A. K. M., *Stochastic Perturbation Methods for Robust Optimization of Simulation Experiments*, M. S. Thesis: The Pennsylvania State University, State College, Pennsylvania, May 2008.

12 Wang, I.-J., J C. Spall, "Stochastic Optimisation with Inequality Constraints Using Simultaneous Perturbations and Penalty Functions," *International Journal of Control*, Vol. 81, No. 8, August 2008, pp. 1232–1238.

13 NERC, *Methods to Model and Calculate Capacity Contributions of Variable Generation for Resource Adequacy Planning*, North American Electric Reliability Corporation, March 2011.

14 Rabiti, C., A. Alfonsi, J. Cogliati, D. Mandelli, R. Kinoshita, S. Sen, *RAVEN User Manual*, INL/EXT-15-34123, March 2015.

[15] Kim, J. S., H. E. Garcia, “Nuclear-Renewable Hybrid Energy System for Reverse Osmosis Desalination Process,” in *Transactions of the American Nuclear Society*, San Antonio, Texas, 2015, pp. 121–124.

[16] Rahimi, A., B. Recht, “Random Features for Large-Scale Kernel Machines,” *Advances in Neural Information Processing Systems 20 (NIPS 2007), Vancouver, B.C., Canada, December 2007.*