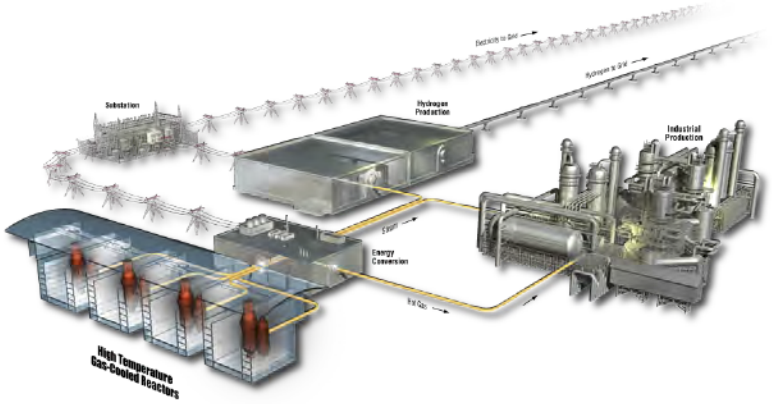
INL/EXT-16-39832
Revision 0

Status on the Development of a Modeling and Simulation Framework for the Economic Assessment of Nuclear Hybrid Energy Systems (FY 16)

Aaron Epiney
Jun Chen
Cristian Rabiti

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

September 2016



INL/EXT-16-39832
Revision 0

Status on the Development of a Modeling and Simulation Framework for the Economic Assessment of Nuclear Hybrid Energy Systems (FY 16)

Aaron Epiney
Jun Chen
Cristian Rabiti

September 2016

Idaho National Laboratory
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

ABSTRACT

Continued 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) 2016. 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, a synthetic time history generator has been developed in RAVEN, which consists of Fourier series and autoregressive moving average model. The former is used to capture the seasonal trend in historical data, while the latter is to characterize the autocorrelation in residue time series (e.g., measurements with seasonal trends subtracted). As demonstration, both synthetic wind speed and grid demand are generated, showing matching statistics with database.

In order to build a design and operations optimizer in RAVEN, a new type of sampler has been developed with highly object-oriented design. In particular, simultaneous perturbation stochastic approximation algorithm is implemented. The optimizer is capable to drive the model to optimize a scalar objective function without constraint in the input space, while the constraints handling is a work in progress and will be implemented to improve the optimization capability.

Furthermore, a simplified cash flow model of the performance of an NHES in the electric market has been developed in Python and used as external model 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.

Finally, an example calculation is performed that shows the integration and proper data passing in RAVEN of the synthetic time history generator, the cash flow model and the optimizer. It has been shown that the developed Python models external to RAVEN are able to communicate with RAVEN and each other through the newly developed RAVEN capability called “EnsembleModel”.

CONTENTS

FIGURES

TABLES

ACRONYMS

AIC	Akaike Information Criterion
ANN	Artificial Neural Network
AR	Autoregressive
ARMA	Autoregressive Moving Average
AR-GARCH	Autoregressive Generalized Autoregressive Conditional Heteroskedasticity
BIC	Bayesian Information Criterion
BNN	Bayesian Neural Network
CDF	Cumulative Distribution Function
ERCOT	Electric Reliability Council of Texas
FCFF	Free Cash Flow to Firm
FDSA	Finite Difference Stochastic Approximation
FFT	Fast Fourier Transform
FMU	Functional Mock-up Unit
FOM	Figure of Merit
FS	Finkelstein-Schafer
FY	Fiscal year
INL	Idaho National Laboratory
KKT	Karush–Kuhn–Tucker
LCOE	Levelized Cost of Electricity
LOLP	Loss of Load Probability
MC	Markov Chain
MLE	Maximum Likelihood Estimation
NHES	Nuclear Hybrid Energy Systems
NN	Neural Network
NPV	Net Present Value
NREL	National Renewable Energy Laboratory
NWP	Numeric Weather Prediction
PSD	Power Spectrum Density
RAVEN	Risk Analysis Virtue ENvironment
SPSA	Simultaneous Perturbation Stochastic Approximation
WACC	Weighted Average Cost of Capital
XML	eXtensible Markup Language

Status on the Development of a Modeling and Simulation Framework for the Economic Assessment of Nuclear Hybrid Energy Systems (FY 16)

1. INTRODUCTION

This report provides the current status of development of a software framework to perform economic evaluation of NHES (Nuclear Hybrid Energy Systems). The report highlights the fiscal year (FY) 2016 developments, which focus on developments for the software framework. Progresses on the Modelica models as well as a description of the developed software development infrastructure are provided in separate reports.

2. BACKGROUND

One of the goals of the HYBRID modeling and simulation project is to assess the economic viability of hybrid systems in a market that contains renewable energy sources like wind. The hybrid system would include a nuclear reactor that not only generates electricity, but also provides heat/electricity to another plant that produces by-products, like hydrogen or desalinated water. The idea is that the possibility of selling non-electric energy cushions (at least part of) the volatility introduced by the renewable energy sources (Rabiti et al. 2015 report [1]).

As largely discussed in Rabiti et al. 2015 [1], the problem to solve is to find the optimal configuration of an NHES that will minimize the cost of electricity, 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. The introduction of such constraints leads to the calculation of an effective cost of electricity that differs from the levelized cost of electricity (LCOE) since the cost of electricity is computed a posteriori to account for the effective utilization of each subsystem. This document refers to the cost of electricity as the cost of electricity incurred when the coverage of the demand is imposed.

The system that is studied is modular and made of an assembly of components. For example, a system could contain a nuclear reactor, a gas turbine, a battery, a by-product production subsystem and, possibly, renewables. This system could correspond to the size of a balance area, but in theory any size of system is imaginable. The system is modeled in the 'Modelica/Dymola' language [2, 3]. As mentioned in the introduction, this report focuses on the current status of the simulation framework, while the different component models are described in separate reports.

To assess the economics of the system, an optimization procedure will be performed to obtain a set of parameters, which defines the configuration of the NHES to find the minimal cost of electricity. Figure 1 shows a diagram of the software framework for the NHES modeling and optimization. As one can see, the statistics and optimization code RAVEN is used as a driver for the whole problem. RAVEN is running the optimization, i.e. RAVEN changes the input parameters in the system model, provides the needed time histories for demand, wind, etc., runs the Modelica system model, collects output from Modelica and assesses the next optimization step. As mentioned, the figure of merit (FOM) for the optimization is the cost of electricity. The Modelica output is used in a simple cash flow analysis that will

reveal the cost of electricity. This cash flow analysis module is loaded by RAVEN as an "external model" [4]. This document describes first the "synthetic time history generator" (Chapter 3) developed as a reduced order model (ROM) in RAVEN. This time history generator is used to provide generic time histories for electricity demand as well as different renewables availabilities like wind or sun histories to the Modelica model. Next, the developed optimization routine is described (Chapter 4). The document further describes the cash flow calculation and highlights the needed information passing between the Modelica model and the cash flow calculation (Chapter 5). Finally, an example is shown that demonstrates the integration of the 3 components, namely, "time history generator", "optimization", and "cash flow calculation" (Chapter 6).

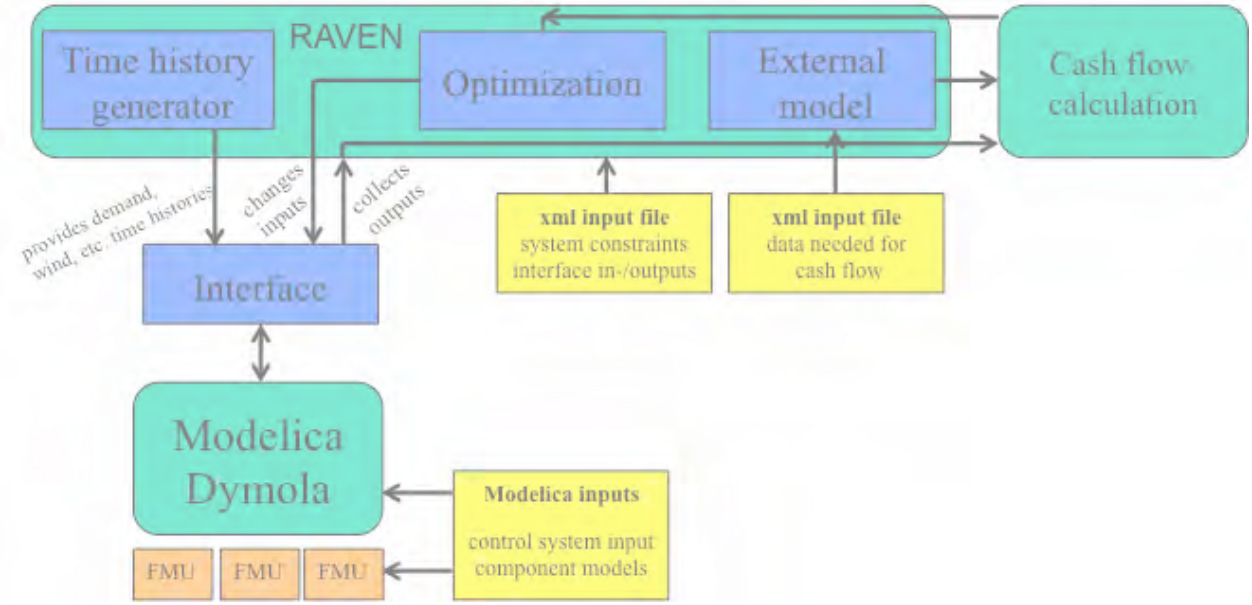


Figure 1. HYBRID system modeling and optimization framework

3. SYNTHETIC TIME SERIES GENERATION BY ARMA

This chapter briefly discusses the procedure to generate synthetic scenarios for renewable generations and grid loads. The generated time series are shown to statistically conform to the actual measurement but possess different temporal profiles. In particular, a combined model with Fourier series and autoregressive moving average (ARMA) is utilized to de-trend the yearly measurement and to characterize the autocorrelation of the residues. The trained model is then able to generate synthetic time series. The synthetic data generation consists of generating independent white noise for each time step, utilizing ARMA model to compute residues for each time step, and finally adding the Fourier series representing seasonal trends. The model is implemented as part of Risk Analysis Virtual ENvironment (RAVEN) developed at Idaho National Laboratory (INL) [4]. To validate the trained model and the corresponding synthetic time histories, key statistics generated from actual database as well as that from synthetic data are compared, including mean, variance, quantiles, and empirical cumulative distribution function.

3.1 Related Literature

3.1.1 Point Forecast vs. Scenario Generation

Two types of research have been conducted in literature, point forecast vs. scenario generation. The former focuses on predicting the exact future value in time series, i.e., predicting exact waveform, while scenario synthesis generates waveforms with equivalent statistical characteristics (mean, volatility, autocorrelation, etc.) with those from data. The time span for point forecast can be very short term (from seconds to 30 minutes), short term (30 minutes to 6 hours), medium term (6 hours to 1 day) or long term (1 day to 1 week or more), while the time span for scenario generation is usually long term (1 week to 1 year).

3.1.2 Wind Speed Prediction

Even though this chapter doesn't focus on point forecast, existing work on this area are summarized below for the sake of completeness. Point forecast techniques are categorized into physical approach and statistical approaches. Physical approaches use meteorological data, which are good for long-term prediction, but lack of short-term accuracy. Statistical approaches provide accurate short-term results, but their reliability in long-term prediction is questionable. Statistical approaches are further categorized into artificial neural network (ANN) and time-series models. The later includes, for example, autoregressive models. Both ANN and time series models can be trained over historical data and metrological data (e.g., NWP [numeric weather prediction]).

Some recent research combines a variety of techniques for more accurate prediction. Reference [5] combines wavelet transform with an AR model. Firstly, the wavelet method is applied to decompose original time series into a number of different sub-series. Then the AR model is built for each sub-series for prediction in its domain. Finally aggregating the prediction in each sub-series provides the final forecasting. Reference [6, 7] use Kalman filter to improve available NWP forecast on wind speed.

Let m_t be the NWP forecast, and y_t be its error (the difference between m_t and the real value). The relation between y_t and m_t is assumed to be: $y_i = x_{0,t} + x_{1,t}m_t + x_{2,t}m_t^2 + \cdots + x_{n,t}m_t^n + v_t$. Then the goal is to estimate parameter X_t to fine tune future NWP forecast y_t. The state and output equations become:

$$X_t = X_{t-1} + w_t$$

$$y_t = H_t x_t + v_t$$

where H_t is $[1, m_t, m_t^2, \ldots, m_t^n]$. Kalman filter is readily applied here.

Another approach in point forecast of wind speed includes the Mycielski approach and Grey model. For example, [8] predicts the next value in currently ongoing random process by the longest repeating data chain that has shown up in the past data sequence. [9] demonstrates the Grey model for wind speed prediction.

Let X be the time series, then the Grey model is expressed as $dX/dt + aX = b$. The only two parameters a and b are estimated from available historical data, and the prediction of next value is given by $\hat{X}(i+1) = \left(X(1) - \frac{b}{a}\right)e^{-ai} + \frac{b}{a}$.

3.1.3 Wind Speed Scenario Generation

The synthetic data generation has been studied in the literature [10], where different synthesis algorithms have been proposed. The simplest yet intuitive algorithm is the empirical approach, which multiplies the empirical time series by a constant. Another approach combined empirical approach with the ARMA model, where the ARMA model is not used to generate directly the data of interests, but the noise term that will later be added to the empirical time series. For example, [11] uses ARMA to generate prediction errors, and adds the sample prediction errors to the historical data. The ARMA model itself has also been proposed to generate wind speed scenarios. For example, [12,13,14] use ARMA or AR models to fit available wind speed data, and then use the resulting ARMA model and sampled white noise to generate scenarios. The original time series may need to be transformed into Gaussian distribution by its marginal distribution (either empirical or fit to parametric distribution) to fit the assumption of ARMA. Reference [13] further normalizes the transformed time series with respect to each hour of each month. Reference [15] uses AR-GARCH (autoregressive generalized autoregressive conditional heteroskedasticity) for wind speed prediction, which allows the regression of both mean and variance. Seasonal terms were added in the mean regression to account for seasonal effect. The prediction output, in terms of the mean and variance of the wind speed/power, can then be sampled to generate both sample paths and point forecast.

Application of the Gaussian process and neural network has also been found in literature for scenario generation. Reference [16] uses Gaussian process for point prediction, which takes metrological data [16] as input and wind prediction as output. Reference [17] applied similar methodology over historical wind speed data. After choosing a proper covariance function, the hyper-parameters can be estimated by maximum likelihood function. Note that these works focus on point prediction instead of scenario generation, but since the estimated model is a distribution, the sample path can also be synthesized.

Authors in [18] use factor analysis for scenario synthesis, a statistical method to describe variability among observed correlated variables in terms of a potentially lower number of unobserved variables called factors. Let X be the observed data, its linear representation over factors can be written as $X = LF + E$, where L is called loading matrix and F is called factors. The dimension of F represents the number of factors. E is random noise with covariance Φ. Values for L, Φ and F can be found by fitting the observation covariance $S = cov(X)$ into $LL_T + \Phi$ and F is estimated to be $XS^{-1}L$. In wind application, data needs to be normalized before applying factor analysis. Sample path generations follow as simulating the random noise E to generate randomized X and de-normalizing the data. Reference [19] computes power spectrum density (PSD) through measurement data or through AR(MA) model of the data, predicts PSD based on future capacity, and generates sample waveform by inverse FFT of predicted PSD. When doing so, phase angles are determined by a genetic algorithm to fit certain statistical characteristics including: uniform distributed phase angle, minute-to-minute difference distributed as Laplace, ramping distribution, etc.

3.1.4 Electric Load Scenario Generation

Reference [20] uses AR model to fit available sub-hour load data. The linear and Fourier terms are used to fit the seasonal trend, with remaining irregular load modeled by an AR model. Reference [21] uses ARMA to fit the residues that result from de-trending with seasonal latent variables. Reference [22] also decomposes the load data into deterministic seasonal trend part and irregular part. Both parts are used to train NN, one for each, which are used to generate the forecast load. NN-based methods can also be found in [23,24,25]. Reference [25] combines wavelet transform with NN approach. First use the wavelet method to decompose original time series into a number of different sub-series. Next, the NN is trained for every sub-series. Finally aggregating prediction in each sub-series obtains the final forecasting. Reference [26] uses a Bayesian belief network to improve available load forecasting. The historical data on forecasting errors is analyzed and used to fit the BNN, which is used to adjust the future forecast.

3.2 Theoretical Foundation

3.2.1 ARMA Model and Identification

Autoregressive moving average (ARMA) model provides a mathematical framework to characterize stationary processes by imposing a linear dependence among the variables and a series of white noise [12]. An ARMA model with orders p and q, often referred to as ARMA(p,q), is given as [27]:

$$x_t = \sum_{i=1}^{p} \phi_i x_{t-i} + \alpha_t + \sum_{j}^{q} \theta_j \alpha_{t-j} \tag{1}$$

where the process variable x is a vector of dimension n, and parameters φ_i for $i = 1,\ldots,p$ and θ_j for $j = 1,\ldots,q$ are both n by n matrices. The noise term α is usually assumed to be white noise. When q = 0, the model reduces to autoregressive (AR) model; when p = 0, it is called moving average (MA) model.

Given an ARMA(p,q) model, identification of its parameters φ_i 's and θ_j 's can be done by either maximum likelihood estimation or non-linear least square estimation. Below is an overview of the parameter estimation process. A more detailed treatment of time series and ARMA model can be found in [27].

Suppose there are T number of measurements of the process variable x, denoted as $x_1, x_2, \ldots, x_T$. Denote the covariance of α to be Σ, and

$$\Sigma = \begin{bmatrix} \sigma_1^2 & & & \\ & \sigma_2^2 & & \\ & & \ddots & \\ & & & \sigma_n^2 \end{bmatrix} \tag{2}$$

Define the parameters to be estimated as $\eta := (\varphi_1,\ldots,\varphi_p, \theta_1,..,\theta_q, \sigma_1^2,\ldots,\sigma_n^2)$. Then the likelihood function can be written as:

$$L(\eta) = \prod_{t=1}^{T} \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp\left(-\frac{1}{2} \hat{\alpha}_t' \Sigma^{-1} \hat{\alpha}_t\right) \tag{3}$$

where the estimation of the error term at time t, i.e., $\hat{\alpha}_t$ is recursively computed as:

$$\hat{\alpha}_1 := x_1 \tag{4}$$

$$\hat{\alpha}_2 := x_2 - (\phi_1 x_1 + \theta_1 \hat{\alpha}_1) \tag{5}$$

$$\hat{\alpha}_3 := x_3 - (\phi_1 x_1 + \phi_2 x_2 + \theta_1 \hat{\alpha}_1 + \theta_2 \hat{\alpha}_2) \tag{6}$$

$$\ldots$$

$$\hat{\alpha}_t := x_t - \left(\sum_{i=1}^{\min(p,t-1)} \phi_i x_{t-i} + \sum_{j=1}^{\min(q,t-1)} \theta_j \hat{\alpha}_{t-j}\right) \tag{7}$$

Maximizing the likelihood function $L(\eta)$ is same as maximizing $\log L(\eta)$, which can be expressed as:

$$\log L(\eta) = -\frac{nT}{2}\log(2\pi) - \frac{T}{2}\log|\Sigma| - \frac{1}{2}\sum_{t=1}^{T}\hat{\alpha}_t'\Sigma^{-1}\hat{\alpha}_t \quad (8)$$

The MLE of η is given by $\hat{\eta}$ that maximize the above log-likelihood function, i.e.,

$$\hat{\eta} \coloneqq \arg\max \log L(\eta) \quad (9)$$

Given observed data, the parameters p and q are chosen as following. Though larger values of p and q will generally result in a better fitting of the data, such "*better fit*" can be a result of overfitting. To balance between good data fitting and prevention of overfitting, two selection criteria are widely used, namely Akaike information criterion (AIC) and Bayesian information criterion (BIC), given as:

$$AIC(p,q) = \log(\hat{\sigma}^2) + \frac{2(p+q)}{T} \quad (10)$$

$$BIC(p,q) = \log(\hat{\sigma}^2) + \frac{\log T(p+q)}{T} \quad (11)$$

where $\hat{\sigma}^2$ is the determinant of $\Sigma^{\mathrm{T}}\Sigma$. It is not hard to see that the first term on the right hand side captures how the model fits to the data, while the second term penalizes larger model to prevent overfitting. Note that usually BIC penalizes large models more than AIC. In this paper, BIC is chosen for model selection.

3.2.2 Seasonal Trend and Normality Transform

Two assumptions of ARMA limit its usage to model data. One is that the process is stationary, and the other being that the process variable x shall be normally distributed. Regarding the stationarity, seasonal ARMA model is used to mitigate the difficulty. In other words, the data is first "detrended" by a combination of Fourier series, and the residues between original database and the Fourier series are then used to train the ARMA model. The Fourier series used in this paper is given as:

$$F_t \coloneqq \sum_k \{a_k \sin(2\pi f_k t) + b_k \cos(2\pi f_k t)\} \quad (12)$$

The set of frequency f_k are user-defined parameters, and the coefficient $\{a_k\}$ and $\{b_k\}$ can be estimated by linear regression.

In general, renewable source and load profile do not satisfy the normality assumption, even after seasonal trends being extracted. However, this problem can be moderately mitigated by properly transforming the data so that the transformed data has Gaussian property [12,13,27]. Define a new stochastic process y, which has a standard normal distribution and is defined as:

$$y_t \coloneqq \Phi^{-1}[f(x_t - F_t)] \quad (13)$$

where f is the empirical cumulative distribution function (CDF) of the residues and Φ is the CDF function of the standard normal distribution.

Note that the above procedure removes the seasonal trends from the original data x and transform the residues into stochastic process y that is normally distributed. The transformed data is then used to train the ARMA model. The trained ARMA model can then be used to simulate process y, which is in turn used to generate the scenario by an inverse transformation, as following:

$$x_t \coloneqq f^{-1}[\Phi(y_t)] + F_t \tag{14}$$

3.2.3 Data Pre-processing

The database used in this study, both for renewable resource and load profile, generally spans over multiple years. To train an ARMA model, one-year data is enough to achieve parameter convergence. However, to ensure that the trained model possesses the typical statistical characteristics from the database, a simply pre-processing procedure is applied to translate multiple-year data into "typical"-year data that spans exactly one year.

The pre-processing is a simplified procedure of [28], which selects individual months from different years in the database. For example, all the Januarys are examined and the one determined as most typical would be selected (see below). This procedure continues until all the months of the year are examined and then the twelve selected typical months are concatenated to form a complete typical year. To select a typical month, the empirical CDF of the interested variables (wind speed, load, etc) of the entire multi-year dataset is computed, termed as long term CDF. Then the CDF over each candidate month is also computed, and the typical month is selected based on the Finkelstein-Schafer (FS) statistics [29] between the candidate month's CDF and the long term CDF. The FS statistics is defined as following:

$$FS = \sum_{x_i} FS_{x_i} \tag{15}$$

$$FS_{x_i} = \frac{1}{N}\sum_{n=1}^{N} \delta_n \tag{16}$$

where N is the number of value reading in the empirical CDF and δ_n is the absolute difference between the long term CDF and the candidate month's CDF at value xi. The candidate month with minimal FS statistics will be selected. Such process repeats for every month of the year, and then all the typical month will be concatenated to yield a complete year.

3.2.4 Algorithm Implementation

The above procedure is implemented in RAVEN with Python programming language. The implemented flow diagram in RAVEN is shown in Figure 2.

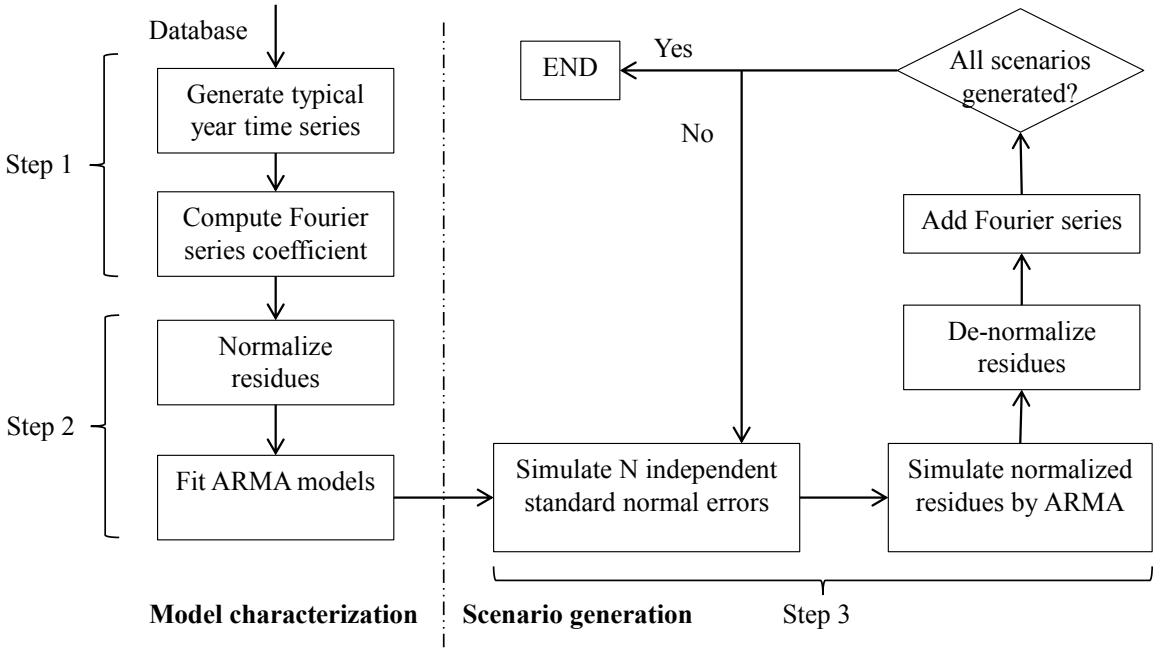


Figure 2. Implementation flow diagram in RAVEN

3.3 Results

3.3.1 Wind Speed Scenario

The wind speed database used to train the model is obtained from NREL, which consists of three years data, namely the years of 2004, 2005, and 2006. The typical year data is formed by the procedure discussed in Section 3.2, and is shown in Figure 3(a). Furthermore, Table 1 shows the constitution of the typical year data. Seasonal trends extracted from typical year data and modeled as Fourier series is given in Figure 3(b).

Figure 4 plots the synthetic wind speed scenario and the actual database for selected 7 days. As can be seen, the synthetic scenario and actual database exhibit similar dynamics and volatility. Furthermore, Table 2 compares several key statistics (mean, standard deviation, etc) between the synthetic and actual data, showing identical statistics between the synthetic scenarios and the actual database. The Quantile-Quantile plot (qq-plot) between the synthetic and actual database is given in Figure 5, while Figure 6 compares the CDF of synthetic and actual database, both suggesting good match between synthetic and actual wind speed scenarios. These results suggest that the presented model can produce synthetic wind speed scenario with same statistical characteristics as the actual database.

Furthermore, to demonstrate the benefit of the synthetic scenarios, Figure 7 plots the actual wind speed with 100 synthetic scenarios, where each synthetic scenario possesses very different time profile from the actual database. Considering that the synthetic wind speed data possess same statistic characteristics with the database while having different temporal profiles, they can be used for Monte Carlo and risk analysis of energy integration systems, while avoiding bias introduced by using a same database.

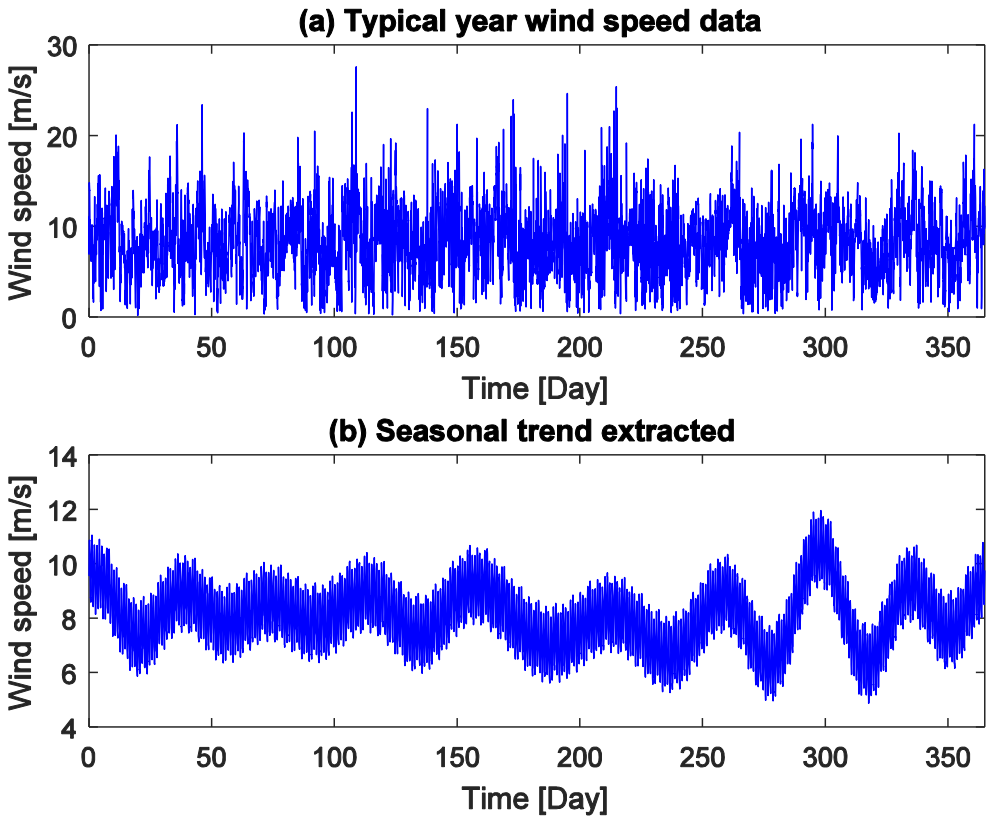


Figure 3. Wind speed data used to train the model; (a) Typical year data; (b) Seasonal trend extracted from typical year data

Table 1. Constitution of Typical Year Data

Month	From Dataset of	Month	From Dataset of
January	2005	July	2006
February	2006	August	2006
March	2004	September	2004
April	2004	October	2004
May	2006	November	2004
June	2006	December	2005

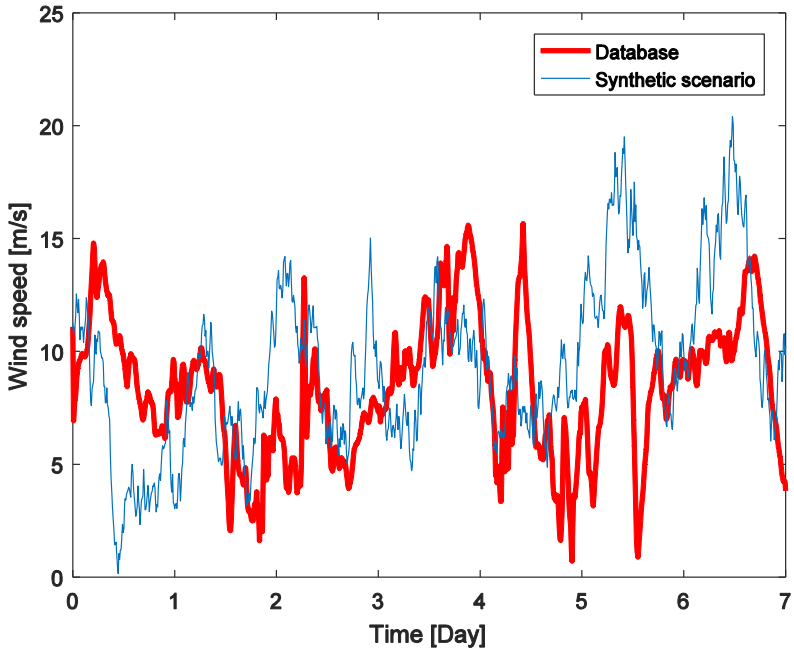


Figure 4. Synthetic wind speed scenario and the actual database for selected 7 days

Table 2. Comparison between Synthetic and Actual Data

Statistics	Database	Synthetic
Mean (wind speed)	8.078	8.088
Standard deviation (wind speed)	3.392	3.372
Mean (step to step difference)	0	0
(Step to step difference)	0.659	0.642

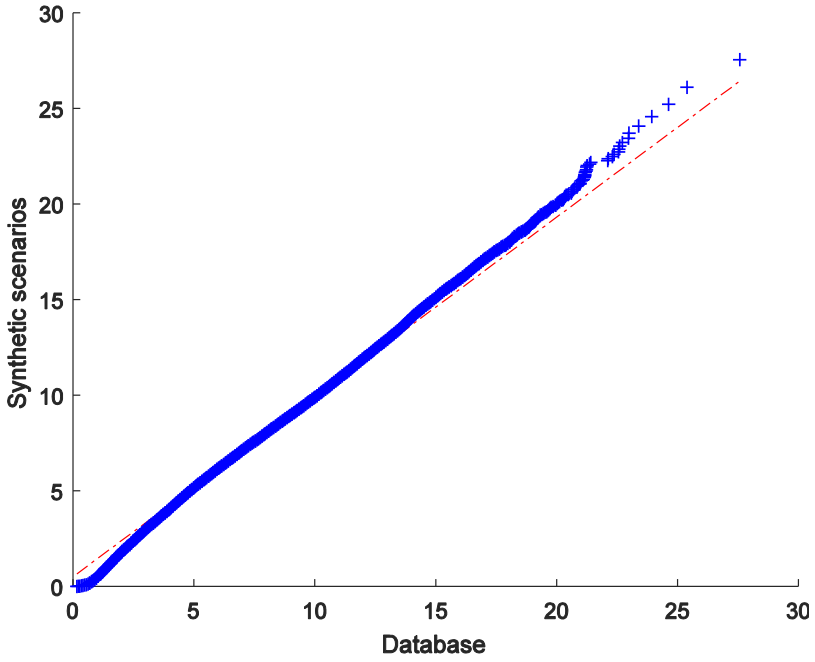


Figure 5. Quantile-Quantile plot (qq-plot) between the synthetic and actual database

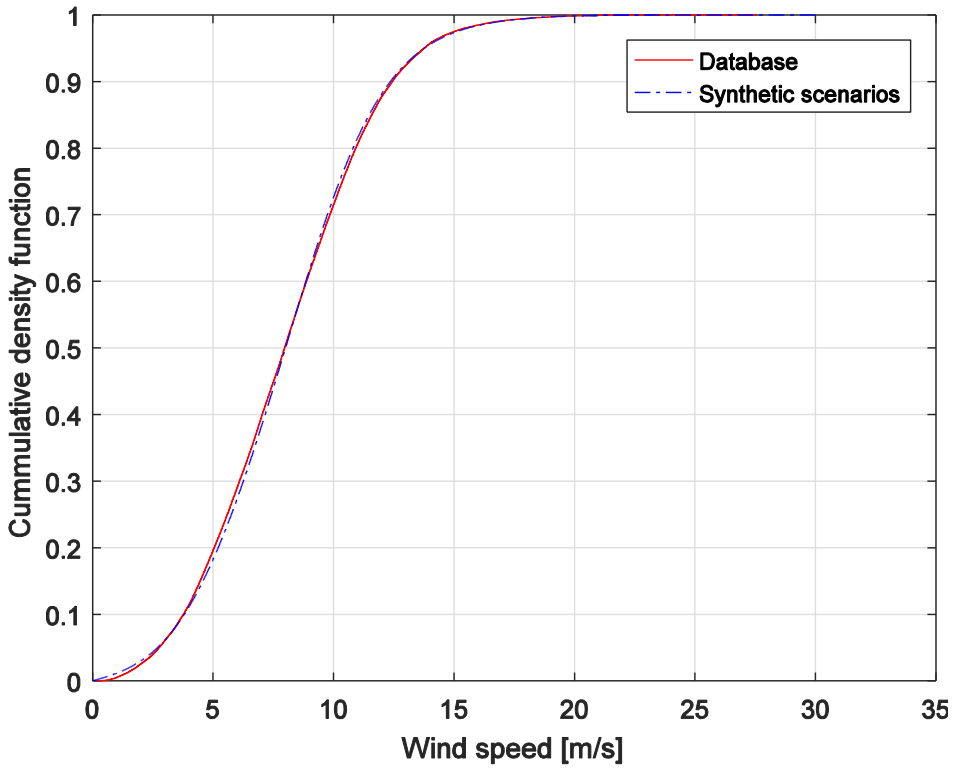


Figure 6. CDF of synthetic and actual database

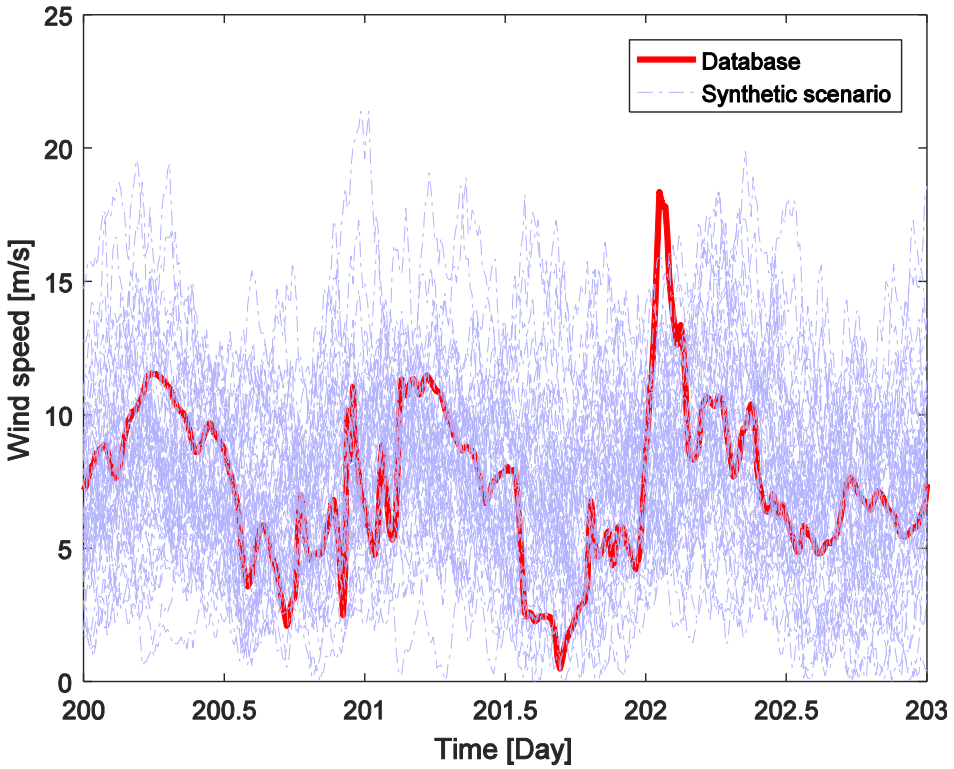


Figure 7. 100 synthetic scenarios and actual database for 3 days

3.3.2 Grid Load Scenario

The grid load database used to train the model is obtained from ERCOT. Figure 8 plots the synthetic load scenario and the actual database for selected 7 days. As can be seen, the synthetic scenario and actual database exhibit similar dynamics and volatility. Furthermore, Table 3 compares several key statistics (mean, standard deviation, etc) between the synthetic and actual data, showing identical statistics between the synthetic scenarios and the actual database. Finally, Figure 9 compares the CDF of synthetic and actual database, suggesting good match between synthetic and actual wind speed scenarios. These results suggest that the presented model can produce synthetic load scenario with same statistical characteristics as the actual database.

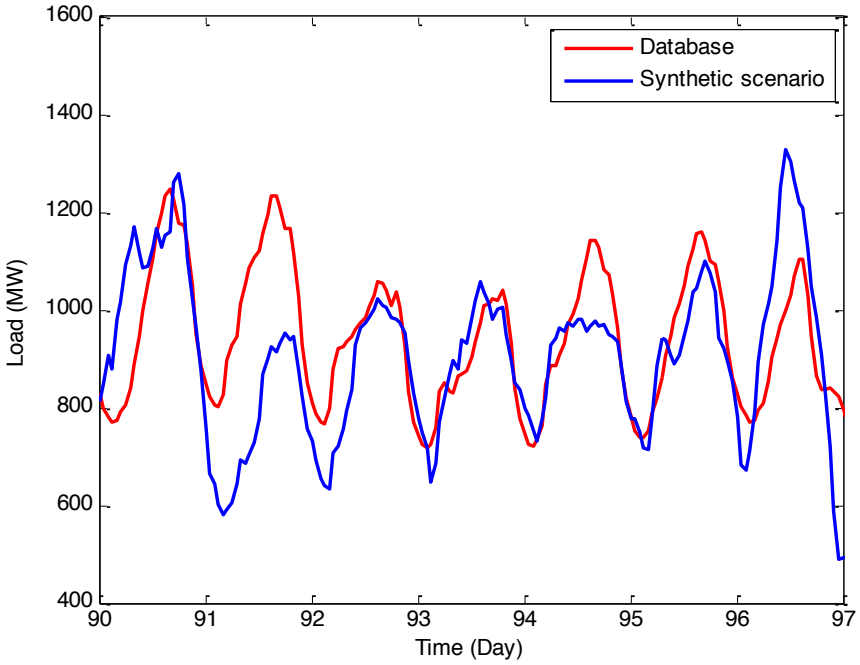


Figure 8. Synthetic load scenario and the actual database for selected 7 days

Table 3. Comparison between Synthetic and Actual Data

Statistics	Database	Synthetic
Mean (load)	1102.3	1103.4
Standard deviation (load)	222.2	223.8
Mean (step to step difference)	0	0
(step to step difference)	48.4	54.2

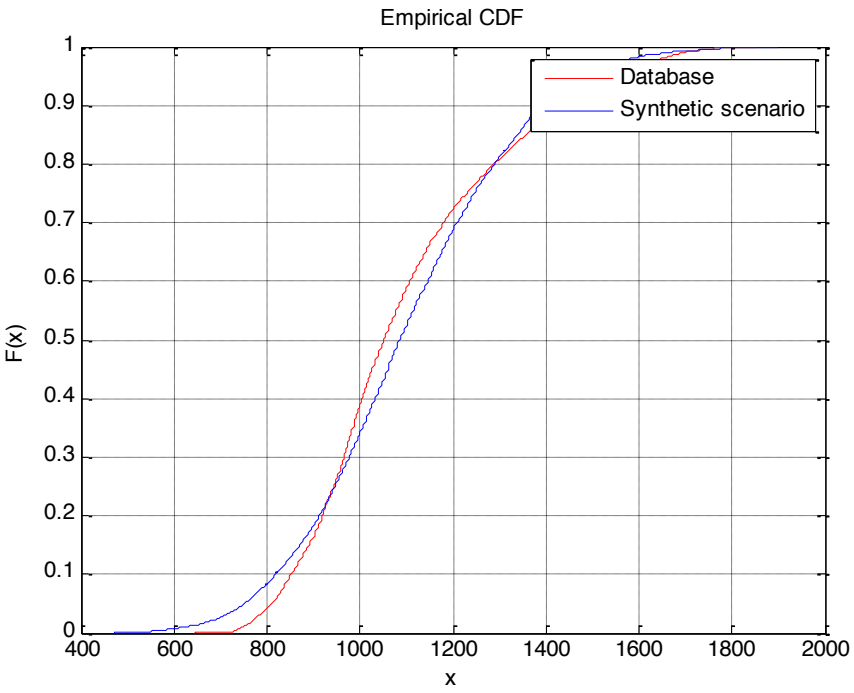


Figure 9. CDF of synthetic and actual database

3.4 Future Work on Generation of Synthetic Time Series

The work so far conducted on the generation of synthetic time series has been successful and it provides a solid support to start the analysis of the stochastic behavior of hybrid systems. This approach allows computing in average optimization so that the final result of the optimization is not bounded to a specific scenario (i.e. the optimization is not only for one specific year). A draw back of the process is that, when multiple years are collapsed in one, a smoothing of the volatility of the time history might happen leading to the diminishing of rare events frequency. Those rare events are the one relevant to the reliability of the grid and therefore this could be a bias to be addressed to ensure that load is matched with the required reliability (above required Loss Of Load Probability). A possible solution currently being considered would be to train the ARMA component over multiple years. This would ensure the removal of the bias.

Once the bias is removed the problem to run enough simulations to ensure that LOLP is below the prescribed threshold (usually below 10^{-5}) could be still challenging. Variance reduction techniques such as weighted windows could be possibly implemented to increase the likelihood of rare events.

4. OPTIMIZATION IN RAVEN

This chapter briefly discusses the optimization capability in RAVEN. Currently, RAVEN has two types of samplers, e.g., statistical sampler and optimizer. The difference between these two is that the latter does not require sampling over a distribution, although certain specific optimizers may utilize stochastic approach to locate the optimality. Figure 10 shows the structure of the optimization workflow. The currently available optimizer is RAVEN is gradient based[a] and stochastic, namely, Simultaneous Perturbation Stochastic Approximation (SPSA).

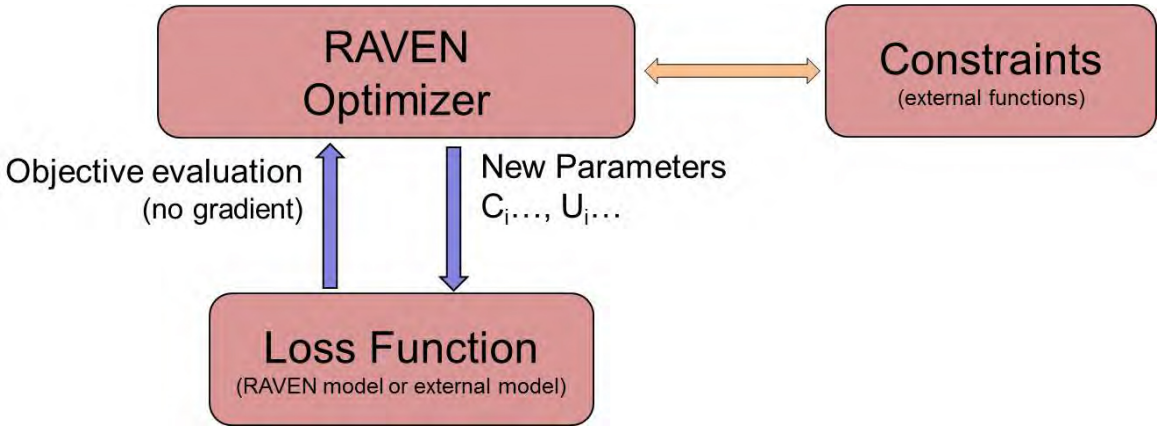


Figure 10. RAVEN optimization workflow

4.1 Simultaneous Perturbation Stochastic Approximation (SPSA)

4.1.1 Theoretical Foundation

This section briefly describes the theorems behind the SPSA algorithm. Details can be found in reference [30, 31, 32].

Let $L(\theta)$ be a differentiable loss function over input θ, which is a n-dimensional vector. The evaluation of $L(\theta)$ is assumed to be obtainable for various values of θ and can be done through the model evaluation of RAVEN. As typical in optimization problem, one of the necessary conditions for optimality is that θ^* should satisfy $\frac{\partial L(\theta^*)}{\partial \theta^*} = 0$. Furthermore, it is assumed that the evaluation of $\frac{\partial L(\theta)}{\partial \theta}$ is not available and needs to be estimated by the optimizer. Let $g(\theta_k)$ be the estimate of the gradient $\frac{\partial L(\theta)}{\partial \theta}$ at the iteration θ_k based on the previous evaluation of loss function, then the general form of the iterative update of θ is then given by:

$$\theta_{k+1} = \theta_k - a_k g(\theta_k) \tag{17}$$

[a] The term "gradient based" is used to refer to the fact that the optimizer will utilize gradient information to compute the new inputs. Since the SPSA algorithm will estimate the gradient instead of relying the model to provide such information, it may also be referred to as "gradient free" in literature. The SPSA algorithm is currently implemented as a sub-class of gradient based optimizers in RAVEN.

where the gain sequence a_k is updated as:

$$a_k = \frac{a}{(A+k)^{\alpha}} \tag{18}$$

where a, A, and α are parameters that can be specified by users through RAVEN input file. To estimate the gradient $g(\theta_k)$, the algorithm randomly generates a perturbation vector, and evaluate the loss functions at both θ_k+*perturbation* and θ_k-*perturbation*, i.e., $L(\theta_k \pm perturbation)$. This can be done by generating perturbing inputs in the sub-cycle. Denote the perturbation as $\Delta_k = (\Delta_{k,1}, \Delta_{k,2}, \ldots, \Delta_{k,n})$. Then the gradient for the ith element of θ is estimated as:

$$g_i(\theta_k) = \frac{L(\theta_k + c_k\Delta_k) - L(\theta_k - c_k\Delta_k)}{2c_k\Delta_{k,i}} \tag{19}$$

where the gain sequence c_k is updated as:

$$c_k = \frac{c}{k^{\gamma}} \tag{20}$$

where c and γ are parameters that can be specified by users through RAVEN input file.

Note that SPSA requires only two loss function evaluations each iteration for updating θ, regardless of its dimensions. This is different from Finite Difference Stochastic Approximation (FDSA), which requires $2n$ evaluations. Despite this reduction in loss function evaluation, SPSA is shown to be more efficient than FDSA in terms of computational requirements. This feature is especially important when the underlying loss function is very complicated with large number of optimization variables. Since the Modelica model of hybrid energy systems and the associated cash flow model consists of hundreds of optimization variables and require minutes or even hours to perform one loss function evaluation, SPSA is chosen as the optimization engine in this project.

The above algorithm is an unconstrained optimization algorithm. To handle the constraints that are imposed on the optimization variables, the following procedures are proposed to be implemented. Note that it is assumed that the constraints are implicit, meaning RAVEN doesn't have formula for the constraints but only have access to evaluate the satisfaction of certain value θ_k. When new iterative value θ_k does not satisfy the constraints, the following handlings are proposed.

1. Randomly rotate the gradient vector so the update points to the direction bringing as many improvements as possible, as illustrated in Figure 11.
2. To learn the constraints on-the-fly with reduced order modeling capability currently available in RAVEN. Once a high-confident reduced order modeling of constraints is available, it will be incorporated into the loss function via Lagrange multipliers or KKT multipliers.

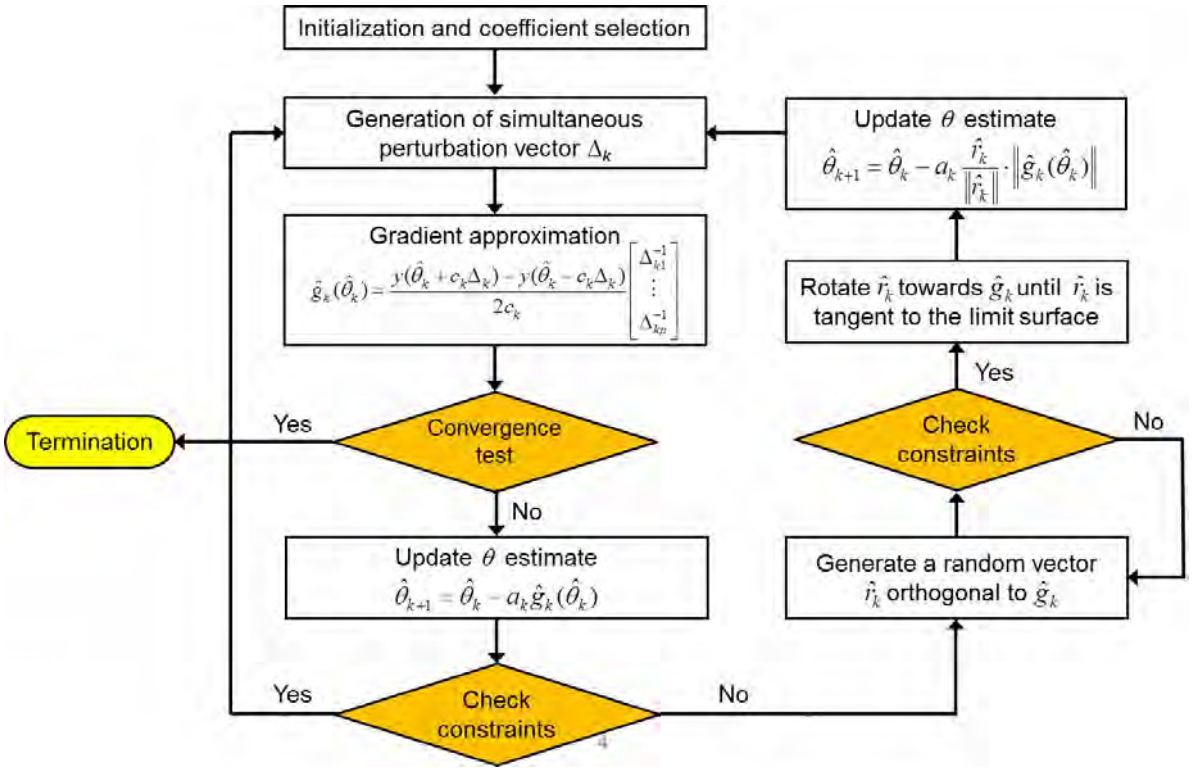


Figure 11. SPSA implementation in RAVEN

4.1.2 Implementation and Input Structure

The workflow of current SPSA implementation in RAVEN is shown in Figure 11, where the constraints handling is a work in progress.

The implementation of SPSA is summarized as follows:

Step 1. Initialize the algorithm and select parameters. Set counter $k=1$, and read parameter values from input file for a, c, A, α, and γ. Default values for these parameters are also given by RAVEN (see next section for details).

Step 2. Generate the perturbation vector. Randomly generate n-dimensional perturbation vector Δ_k. Each component of Δ_k is generated by Monte Carlo from a Bernoulli ±1 distribution with equal probability for +1 and -1.

Step 3. Send the perturbed inputs θ_k+*perturbation* and θ_k-*perturbation* for the model evaluation and collect the evaluated loss function.

Step 4. Approximate the gradient by $g_i(\theta_k) = \frac{L(\theta_k + c_k\Delta_k) - L(\theta_k - c_k\Delta_k)}{2c_k\Delta_{k,i}}$.

Step 5. Update θ_{k+1} by $\theta_{k+1} = \theta_k - a_k g(\theta_k)$

4.2 Examples

This test example is included in RAVEN to validate the SPSA implementation. A simple external model, which is a quadratic function *c* over two input variables *x1* and *x2*, is optimized. In other words

$$c = 10(x_1 - 0.5)^2 + 10(x_2 - 0.5)^2 \tag{21}$$

Therefore, the analytical solution for this minimization problem is $x_1^*=x_2^*=0.5$ with $c^*=0$. RAVEN simulation results are shown in Figure 12 and Figure 13.

Now, consider the case when the measurement of the loss function is contaminated by noise. In other words

$$c = 10(x_1 - 0.5)^2 + 10(x_2 - 0.5)^2 + \delta \tag{22}$$

Here δ is a random normal noise with mean 10 and variance 0.01. Therefore, the analytical solution for this minimization problem is $x_1^*=x_2^*=0.5$ with $c^*=10$. RAVEN simulation results are shown in Figure 14 and Figure 15.

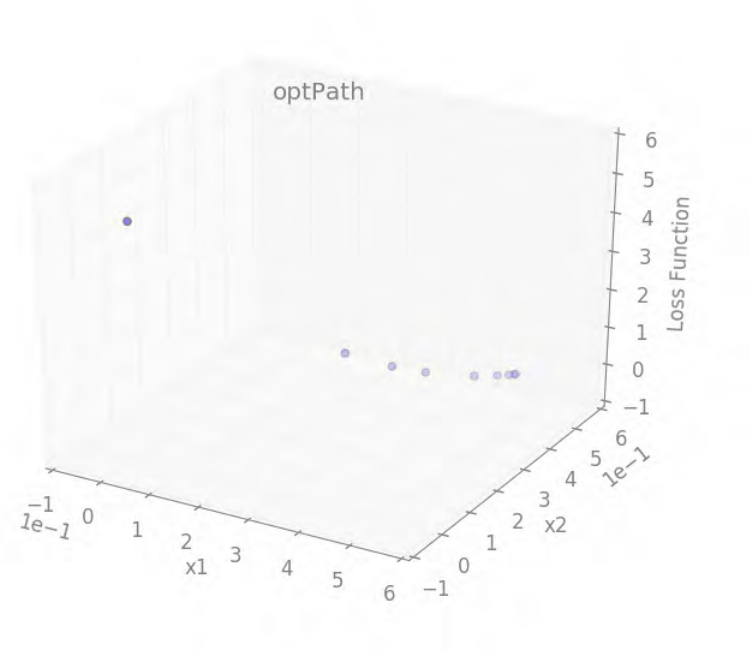


Figure 12. Iteration of optimization variables and loss function values in 3-d

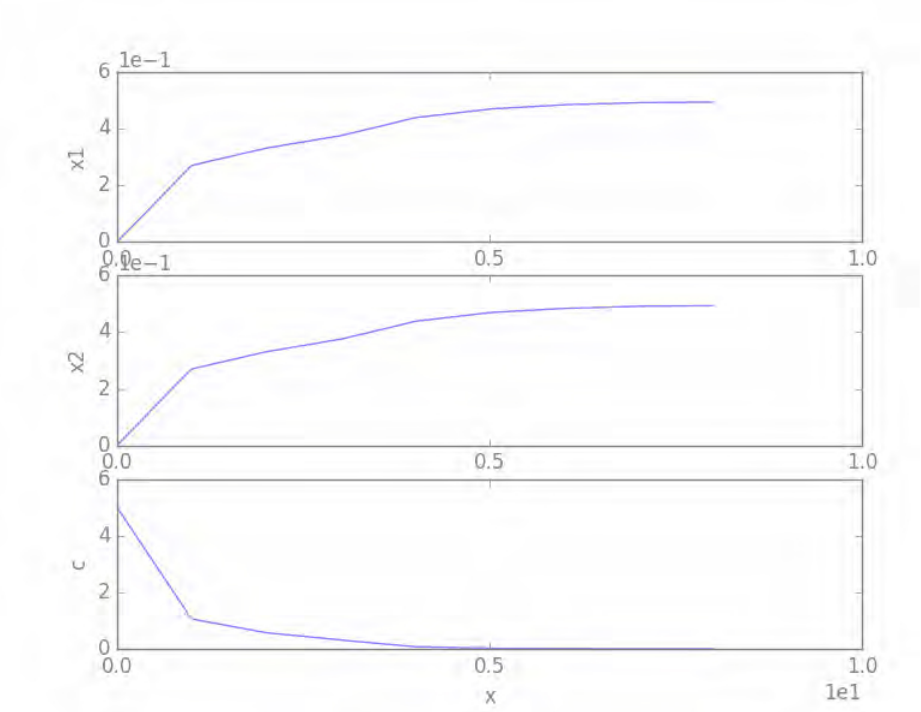


Figure 13. Iteration of optimization variables and loss function values in 2-d

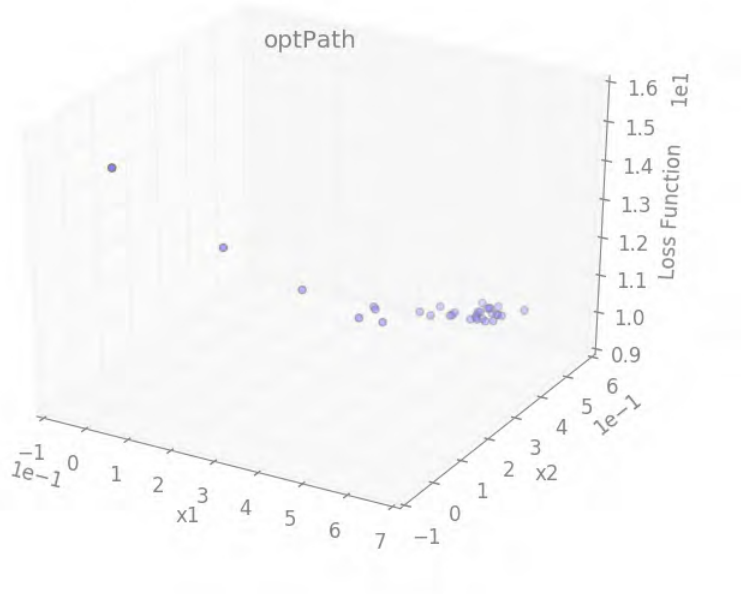


Figure 14. Iteration of optimization variables and loss function values in 3-d

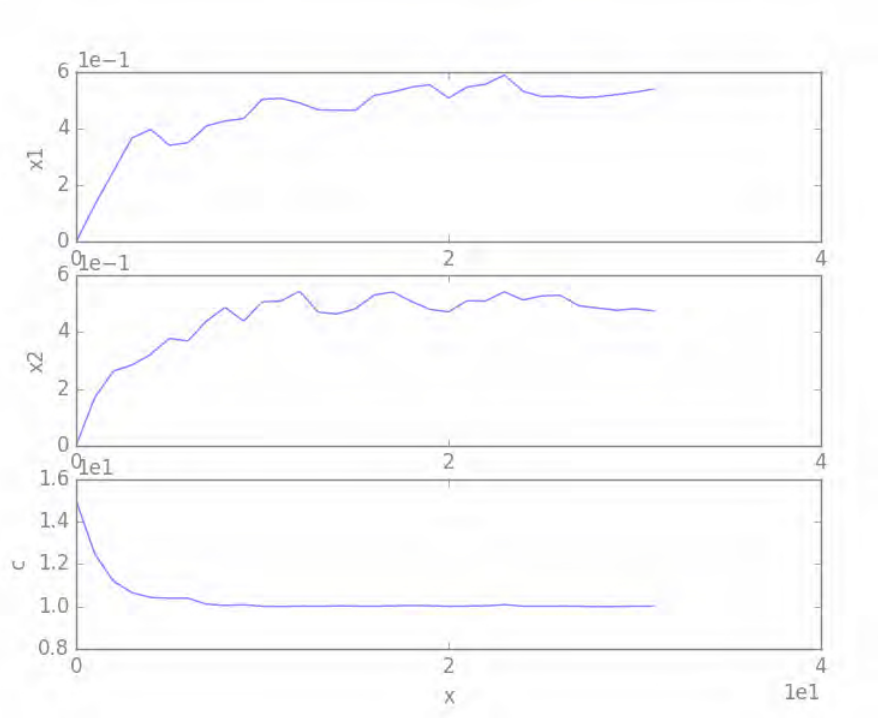


Figure 15. Iteration of optimization variables and loss function values in 2-d

4.3 Future Work on Optimization

With respect to last year, the reliability of the algorithm and its speed have been increased. A dedicated infrastructure with a dedicated input has been developed in RAVEN. The algorithm is part of the stable release of RAVEN now. The treatment of implicit constraints is still challenging, but several approaches are currently being tested. Some more time will be needed to fully test them. The choice of the optimal approach is somehow conditioned by the type of problem to be solved and therefore it will require accumulating experience from the optimization of the real system model. The optimization sampler will also need to be augmented by introducing random initial sampling and redundant estimation of the gradient. These modifications, minimal from the point of view of the code implementation, will allow decreasing the risk to find a relative minimum and/or a minimum representative of a tail of the statistical behavior of the system.

4.4 Sample Inputs and User Manual

4.4.1 Sample Input

```
<?xml version="1.0" ?>
<Simulation verbosity="debug">
 <RunInfo>
    <WorkingDir>SPSA</WorkingDir>
   <Sequence>optimization,optimizationdump</Sequence>
   <batchSize>1</batchSize>
 </RunInfo>

 <Steps>
  <MultiRun name="optimization" pauseAtEnd="true">
   <Input class="DataObjects" type="PointSet">optInput</Input>
   <Model class="Models" type="ExternalModel">optObjModel</Model>
   <Optimizer class="Optimizers" type="SPSA">opt_smp</Optimizer>
   <SolutionExport class="DataObjects" type="PointSet">optData</SolutionExport>
   <Output class="DataObjects" type="PointSet">optOutput</Output>
   <Output class="OutStreams" type="Print">optimizationHistoryDump</Output>
   <Output class="OutStreams" type="Print">mdlDataDump</Output>
  </MultiRun>
  <IOStep name="optimizationdump" pauseAtEnd="true">
   <Input class="DataObjects" type="PointSet">optData</Input>
   <Input class="DataObjects" type="PointSet">optOutput</Input>
   <Output class="OutStreams" type="Print">optimizationHistoryDump</Output>
   <Output class="OutStreams" type="Print">mdlDataDump</Output>
   <Output class="OutStreams" type="Plot">optPath</Output>
     <Output class="OutStreams" type="Plot">plotIteration</Output>
  </IOStep>
 </Steps>

 <DataObjects>
  <PointSet name="optOutput">
   <Input>x1,x2</Input>
   <Output>c</Output>
  </PointSet>
  <PointSet name="optInput">
   <Input>x1,x2</Input>
   <Output>OutputPlaceHolder</Output>
  </PointSet>
  <PointSet name="optData">
   <Input>x1,x2,varsUpdate</Input>
   <Output>c</Output>
  </PointSet>
 </DataObjects>

 <Optimizers>
  <SPSA name="opt_smp">
   <initialization>
```

```
            <limit>300</limit>
            <type>min</type>
            <initialSeed>30</initialSeed>
    </initialization>
    <TargetEvaluation class="DataObjects" type="PointSet">optOutput</TargetEvaluation>
      <convergence>
            <iterationLimit>50</iterationLimit>
            <threshold>1e-3</threshold>
      </convergence>
      <variable name="x1">
      <upperBound>100</upperBound>
            <lowerBound>-100</lowerBound>
            <initial>0</initial>
    </variable>
      <variable name="x2">
      <upperBound>100</upperBound>
            <lowerBound>-100</lowerBound>
            <initial>0</initial>
    </variable>
      <objectVar>c</objectVar>
      <parameter>
            <numGradAvgIterations>3</numGradAvgIterations>
      </parameter>
  </SPSA>
</Optimizers>

<Models>
    <ExternalModel ModuleToLoad="loss_function_quadratic" name="optObjModel" subType="">
            <variables>x1,x2,a1,a2,b1,b2,c</variables>
  </ExternalModel>
</Models>

<OutStreams>
  <Print name="optimizationHistoryDump">
    <type>csv</type>
    <source>optData</source>
  </Print>
  <Print name="mdlDataDump">
      <type>csv</type>
      <source>optOutput</source>
  </Print>

  <Plot dim="3" name="optPath" overwrite="false" verbosity="debug">
      <actions>
        <how>pdf</how>
      </actions>
      <plotSettings>
        <plot>
          <type>scatter</type>
          <x>optData|Input|x1</x>
          <y>optData|Input|x2</y>
```

```
            <z>optData|Input|c</z>
          </plot>
          <xlabel>x1</xlabel>
          <ylabel>x2</ylabel>
          <zlabel>Loss Function</zlabel>
        </plotSettings>
      </Plot>

      <Plot dim="2" name="plotIteration" overwrite="false" verbosity="debug">
        <actions>
          <how>pdf</how>
        </actions>
        <plotSettings>
                        <gridSpace>3 1</gridSpace>
          <plot>
            <type>line</type>
            <x>optData|Output|varsUpdate</x>
            <y>optData|Input|x1</y>
                <interpPointsX>300</interpPointsX>
                <gridLocation><x>0</x><y>0</y></gridLocation>
                <ylabel>x1</ylabel>
          </plot>
          <plot>
            <type>line</type>
            <x>optData|Output|varsUpdate</x>
            <y>optData|Input|x2</y>
                <interpPointsX>300</interpPointsX>
                <gridLocation><x>1</x><y>0</y></gridLocation>
                <ylabel>2</ylabel>
          </plot>
          <plot>
            <type>line</type>
            <x>optData|Output|varsUpdate</x>
            <y>optData|Input|c</y>
                <interpPointsX>300</interpPointsX>
                <gridLocation><x>2</x><y>0</y></gridLocation>
                <ylabel>c</ylabel>
           </plot>
        </plotSettings>
      </Plot>
    </OutStreams>

</Simulation>
```

4.4.2 User Manual

The SPSA optimization approach is one of the optimization strategies that are based on gradient estimation. The main idea is to simultaneously perturb all decision variables in order to estimate the gradient. Consequently a minimal number of two model evaluations are required in order to approximate the gradient. The theory behind SPSA can be found in [30].

The specifications of this optimizer must be defined within a **<SPSA>** XML block. This XML node accepts the following attributes:

- **name**, required string attribute, user-defined name of this optimizer. Note: As for the other objects, this is the name that can be used to refer to this specific entity from other input blocks (xml);

In the **<SPSA>** input block, the user needs to specify the objective variable to be optimized, the decision variables, the DataObject storing previously performed model evaluation, as well as convergence criteria. In addition, the settings for this optimization can be specified in the **<initialization>** and **<parameter>** XML blocks:

- **<initialization**>, *XML node, optional parameter*. In this xml-node, the following xml sub-nodes can be specified:
  - **<limit>**, *integer, optional field*, number of samples to be generated, which is same as the number of model evaluation;
  - **<initialSeed>**, *integer, optional field*, initial seeding of random number generator for stochastic perturbations;
  - **<type>**, *string (case insensitive), optional field*, specifies whether this optimizer performs maximization or minimization. Available options are 'max' and 'min'.
    Default: Min;

- **<TargetEvaluation>**, *XML node, required parameter*, represents the container where the model evaluations are stored. From a practical point of view, this XML node must contain the name of a data object defined in the **<DataObjects>** block (see Section 14). The object here specified must be input as **<Output>** in the Steps that employ this optimization strategy. The **<SPSA>** optimizer accepts "DataObjects" of type "PointSet" only;

- **<objectVar>**, *XML node, required parameter***.** The objective variable to be optimized. This variable must be output of the DataObject specified in **<TargetEvaluation>**.

- **<variable**>, *XML node, required parameter* will specify one attribute:
  - **name**, *required string attribute*, user-defined name of this variable.

The variable specified here must be input of the DataObject specified in **<TargetEvaluation>**. This **<variable>** recognizes the following child nodes:

  - **<upperBound>**, *float, optional field*, the upper bound of this variable;
  - **<lowerBound>**, *float, optional field*, the lower bound of this variable;
  - **<initial>**, *float, optional field*, the initial value for this variable.

- **<convergence>**, *XML node, optional parameter* will specify parameters associated with optimization convergence. This node accepts the following sub-nodes:

- **<iterationLimit>**, *integer, optional field*, user-defined maximum number of optimization iterations.
- **<threshold>**, *float, optional field*, specifies the convergence criteria to determine the optimality. When the change of objective variable in two successive model evaluations is smaller than this pre-specified threshold, the **<SPSA>** optimizer decides optimality and terminates the simulation.
  Default: 1e-3

- **<Parameter>**, *XML node, optional parameter* will accepts the following sub-nodes:
  - **<numGradAvgIterations>**, *integer, optional field* is the number of iterations for gradient estimation.
    Default: 1
  - **<alpha>**, *float, optional field* is a parameter for updating gain sequence for variable update. See [30].
    Default: 0.602
  - **<A>**, *float, optional field* is a parameter for updating gain sequence for variable update. See [30].
    Default: <iterationLimit> divided by 10
  - **<a>**, *float, optional field* is a parameter for updating gain sequence for variable update. See [30].
    Default: 0.16
  - **<gamma>**, *float, optional field* is a parameter updating gain sequence for perturbation. See [30].
    Default: 0.101
  - **<c>**, *float, optional field* is a parameter for updating gain sequence for perturbation. See [30].
    Default: 0.005

5. CASH FLOW ANALYSIS

5.1 Cash Flow Model

As mentioned, the optimization of the Hybrid System tries to minimize the cost of electricity. In order to compute the cost of electricity, a cash flow analysis is performed. The cash flow analysis is used to determine the cost of the electricity that would make the NPV equal to zero. Enforcing therefore a fair economical profit for the NHES in its whole.

The optimization routine seeks various combinations of input variables (according to a defined optimization algorithm) to find the minimum cost of electricity while the system is requested to cope with random synthetic time histories of electricity demand and renewable supply. The analysis does not rely on bidding on marginal cost for every hour, but tries to find the minimal cost to produce a certain amount of electricity with a given time profile.

The cash flow analysis proposed here is an investment analysis based on the Free Cash Flow to the Firm (FCFF) [33, 34, 35]. The reason for choosing the firm point of view is that we consider mostly large companies possibly interested to enter the market. In such cases, it is more logical to consider the debt constantly revolved and not eliminated at the classical 30 years expiration of the loan terms.

The basic assumption of the analysis is that

$$NPV_{FCFF} = 0 \tag{23}$$

where NPV stands for Net Present Value

The NPV is the difference between the present value of cash inflows and the present value of cash outflows for a company. The cash inflows and outflows are balanced out accounting for the different time at which they take place by the discounting factor. A positive NPV indicates that the project offers profit in excess of comparable (risk comparable) market investment. An opportunity cost analysis would highlight that a negative NPV would be equivalent to a positive opportunity cost with respect buying securities in the open market. A positive NPV usually is the condition for a new entrant to appear into the market.

Just to dispel a common misconception, if a company has an NPV of zero, it means that the firm has a positive accounting profit which is proportional to the risk being taken and has an opportunity cost of zero with respect to investing in securities of comparable risk. The cost of electricity that satisfies that criterion is the minimal cost to produce the electricity, since it would correspond to the minimal profit at which a company would enter the market. Expanding the NPV in Equation 23 with its definition, leads to

$$\sum_{y=0}^{N} \frac{FCFF_{R,y}}{(1+WACC_R)^y} = 0 \tag{24}$$

where

N	Total lifetime of the project
WACC	Weighted Average Cost of Capital
FCFF	Free Cash Flow to Firm

The subscript y runs over the years of the lifetime of the project and R indicates that "real" values are considered as opposed to "nominal" values. The difference between "nominal" and "real" cash flow is that the "real" one is adjusted for inflation. One can see that N runs over the whole optimization time, i.e. the optimizer will find the minimum for the cost of electricity considered over the whole optimization time.

From Equation 24, one can see that in order to compute the NPV, the FCFF and the WACC are needed.

5.1.1 $WACC_R$

The Weighted Average Cost of Capital (WACC) can be expressed as

$$WACC_R = \frac{C-E}{C}(1 - tax)r_{RD} + \frac{E}{C}r_{RE} \tag{25}$$

where
- C — Capital
- E — Equity
- tax — Tax rate
- r_{RD} — real cost of debt
- r_{RE} — real cost of equity

The WACC is one of the most influential numbers in the NPV analysis, and, at the same time, very difficult to determine. There are studies aimed to monitor the WACC by the industrial sector but they are build on a posteriori analysis since companies usually are reticent to release their accepted values. As it can be seen in [36] the nominal WACC for the energy sector is very low, due to the low ratio E/C. It is useful to recall the formula to convert the nominal into real discount rates:

$$d_R = \frac{1+d_N}{1+i} - 1 \tag{26}$$

where d_R and d_N are generic discount rates and i is the forecasted inflation rate. Using [36] and assuming 0.8% inflation rate for the reference sector the WACC (real) would be 4.8% which is slightly different from the value which it would be obtained using a linear approximation (4.88%)

5.1.2 $FCFF_{R,y}$

The Free Cash Flow to Firm (FCFF) for year y is given by

$$FCFF_y = \big(R_y - OM - DA_y\big)(1 - tax) + DA_y - \Delta WC_y - CAPEX_y \tag{27}$$

Assuming that all the quantities scale with the inflation, except the depreciation, as:

$$X_y = X_0(1 + i)^y \tag{28}$$

where
- X — Any quantity to be discounted by inflation
- i — Inflation rate

one finds the following $FCFF_{R,y}$ for different years in the project.

Year	$FCFF_{R,y}$
0	$-C$
1	$(R_0 - OM_0)(1 - tax) + DA_1(1 + i)^{-1}tax$
A	$(R_0 - OM_0)(1 - tax) + DA_A(1 + i)^{-A}tax$
A+1	$(R_0 - OM_0)(1 - tax)$
N	$(R_0 - OM_0)(1 - tax) + SV(1 + i)^{-N}$

(29)

where C Capital
R Revenue
OM Operation & maintenance
DA_y Depreciation/amortization
SV Salvage value

As mentioned, the studied system contains multiple components, like a hybrid nuclear reactor, a gas turbine, a battery and some renewables. The current analysis focuses on finding the minimum cost of electricity for the global system. Therefore, individual components (like for example the battery storage or the gas turbine) are allowed to lose money, as long as the global system is profitable. The condition NPV = 0 as described in Equation 23 is therefore evaluated for the whole system as shown in Equation 30 and not for each component individually.

$$\sum_{compo} NPV_{compo} = 0 \tag{30}$$

where compo Sum over all components in the system
NPV Net Present Value

5.2 Information Exchange with Modelica

From the above discussion, the values needed to evaluate the cash flow can be grouped into **INPUTS, VARIABLES, OPTPARAMS** and **OUTPUTS**.

The **OUTPUT** of the cash flow analysis is the "*electricity cost*". The optimization algorithm will try to find the set of VARIABLES and OPTPARAMS that leads to the lowest electricity cost.

- *Electricity revenue.* This is computed from 'electricity cost'*'production' for each component. Here, the 'electricity cost' is the parameter that we want to optimize. The 'production' for each component is obtained from Modelica (see. VARIABLES).

The **INPUTS** are constants. They don't change during the optimization process and will be provided in a separate input file that the cash flow module can read. These INPUTS are:

- *Amortization time.* Since the amortization time can be different for different components, this value is input for each component.
- *Total lifetime of the project.* This is a global parameter, i.e. the same for all components.
- *WACC*. This is a global parameter. It is debatable if, in the future, this parameter should be given by component. It would eventually make sense if the different subsystems of the NHES have different capital to debt ratios and different investment risk grades.
- *Capital cost.* This is computed from 'capital cost/capacity installed'*'capacity installed' for each component. The factor 'capital cost/capacity installed' can be input for each component. The 'capacity installed' for each component is obtained from RAVEN, since this is a parameter of the optimization (OPTPARAM).
- *Non-electric revenue.* This includes all revenue for each component that does not come from electricity, but for example from selling heat. This parameter is assumed to scale with production, i.e. 'non-electric revenue'*'production'. The non-electric revenue can be input for each component, where the production is obtained from Modelica (see VARIABLES).
- *OM_fix.* The fixed operation and maintenance costs are assumed to scale with the installed capacity, i.e. OM_fix = 'OM/capacity installed'*'capacity installed', where 'OM/capacity installed' is input for each component and 'capacity installed' comes from RAVEN, since it's a parameter of the optimization.
- *OM_varia.* The variable operation and maintenance costs are assumed to scale with the production, i.e. 'OM factor'*'production', where the OM factor is input for each component and the production is obtained from Modelica (see VARIABLES). The OM_varia includes all resource consumptions that arte not explicitly treated. For example fuel waste can be included in OM_varia, but fuel consumption for example is explicitly considered (see below).
- *Tax*. The tax rate is a global parameter.
- *Depreciation/amortization*. This is input for each component and each year of the amortization time.
- *Salvage value*. This parameter is input for each component.
- *Inflation*. Inflation is a global parameter.

The **VARIABLES** are values that will come from Modelica. These are:

- *Production for each component for each hour.* Although RAVEN will change the production for the different components during the optimization, these values should be read from Modelica instead of RAVEN, since the demand may not always be met. Some components may return less (or more) production than asked for by RAVEN.
- *Fuel consumption.* The fuel consumption is treated explicitly. It is computed from 'fuel consumption/capacity installed'*'production', where both, the 'fuel consumption/capacity installed' and the 'production' are obtained from Modelica.
- *CO2 production*. The CO2 production is also treated explicitly. Similar to the fuel consumption, the CO2 production is computed from 'CO2 produced/capacity installed'*

'production', where both, the 'CO2 production/capacity installed' and the 'production' are obtained from Modelica.

Finally the **OPTPARAMS** are variables that RAVEN can adjust. They are also needed in the cash flow analysis. The cash flow analysis tool will get these values from RAVEN.

- *Installed capacity for each component*

Considering this discussion, an input file for the Cash Flow Analysis model looks like this:

```
<Cash_Flow verbosity="0"> <!-- "0" all debug output, "1" some output, "100" only errors  -->
    <!-- input for the global variables -->
    <Global>
        <Project_life>40</Project_life > <!-- years -->
        <WACC>0.07</WACC> <!-- %/100 -->
        <tax>0.2</tax> <!-- %/100 -->
        <inflation>0.03</inflation> <!-- %/100 -->
    </Global>
    <!-- component blocks, input for each component in the system -->
    <Component  name="My component 1, e.g. Reactor, gas turbine, etc.">
        <Amort_time>30</Amort_time> <!-- years -->
        <Efficiency>0.35</Efficiency> <!-- to compute capacity(th)=capacity(el)/efficiency -->
        <Capital_per_capacity>5000</Capital_per_capacity> <!-- $/kW(el) -->
        <No_elec_revenue_price>0.5</No_elec_revenue_price> <!-- $/[kW(th)*h] -->
        <OM_per_capacity>3</OM_per_ capacity> <!-- $/kW(el) -->
        <OM_per_production>2</OM_per_production> <!-- $/kW(el+th)h -->
        <DepreciationAmortisation> 0.02 0.03 0.02 0.02 … 0.03</DepreciationAmortisation>
            <!-- %/100 for each year -->
        <Salvage_value>1000000</Salvage_value> <!-- $ -->
    </Component>
    <Component  name="My component 2">
        …
    </Component>
    …
</Cash_Flow>
```

5.3 Implementation of the Cash Flow Analysis as an External Model in RAVEN

The cash flow analysis described above has been implemented using the script language Python. This Python code can then been used as an "external model" in RAVEN. The model contains all the error and input/output handling code as well as the computation of the cost of electricity, the fuel consumption and CO2 production from sampled quantities and inputs passed from RAVEN.

First, the model looks for input variables passed in through RAVEN. The name convention for these variables is ***[component name]_[input]***. According to the VARIABLES and OPTPARAMS lists above, the inputs for each component that are required to come through RAVEN (either sampled by RAVEN or passed to RAVEN from another model or code like Modelica) are the capacity, the electricity and

byproduct production as well as the fuel consumption and CO2 production. The input checker will check if these variables are present for all input components. As an Example for a component named "GasTurbine", the model expects

- GasTurbine_capacity [kW(electric)]
- GasTurbine_productionEL [%(electric)]
- GasTurbine_productionBY [%(thermal)]
- GasTurbine_fuel [kg/kW(electric+thermal)]
- GasTurbine_CO2 [kg/kW(electric+thermal)]

as inputs from RAVEN. One can input as many components as necessary, but all 5 inputs have to be present for each component. Furthermore, the input checker will check if a **<Component>** block for each component passed by RAVEN is present in the additional xml input section. The components "name" attribute has to match the component name passed from RAVEN. At last, the input checker verifies that all needed inputs are present in the xml input for each component as well as in the **<Global>** input. If an error is detected, a corresponding error message is displayed on the screen and the code execution stopped.

If the input checker found all variables and inputs present, the cost of electricity is computed. From the description of the cash flow model above, it can be seen that the cost of electricity only appears in one linear term of the equation, i.e. the revenue "R" in Equation 29. The problem is therefore of the form

$$ELx + NL = 0 \tag{31}$$

where
- x the cost of electricity
- EL includes all the terms of the sum (Equation 24) that include the cost of electricity
- NL includes all the terms of the sum (Equation 24) that do not include the cost of Electricity

The cost of electricity can then simply be computed with

$$x = -NL/EL \tag{32}$$

The Python implementation is as follows. For every component, a loop is executed where the quantities of Equation 29 are summed for all years of the lifetime of the project. In particular the loop evaluates and sums for each component:

Capital C=-([*component name*]_capacity*Capital_per_capacity)

Revenue R= R_el+R_nonel, the sum of the revenue from electricity and byproduct sales
R_el=[*component name*]_productionEL*8760.0*[*component name*]_capacity*(1-tax)
R_nonel=No_elec_revenue_price*[*component name*]_productionBY*8760.0*[*component name*]_capacity/Efficiency*(1-tax)

O&M OM=-((OM_per_capacity*[*component name*]_capacity+OM_per_production*(([*component name*]_productionEL*8760.0)*[*component name*]_capacity+([*component name*]_productionBY*8760.0)*[*component name*]_capacity/Efficiency))*(1-tax))

Depreciation DA=(DepreciationAmortisation*[*component name*]_capacity*Capital_per_capacity)*(1+inflation)^(-y)*tax

Salvage value SV=Salvage_value*(1+inflation)^(-y)

Once the above sums over all years of the project life have been computed for each component, second and third loops sum the values for all the components. These loops also do the division with the WACC (see Equation 24). Two terms are evaluated: One including all contributions that include the cost of electricity ("EL" in Equation 31) and one the rest ("NL" in Equation 31). The loops are as follows:

- *1st loop*: for all components, EL (include the cost of electricity) and NL (does not include the cost of electricity) are summed for all years.
  EL=R_el/WACC, where WACC for year y = (1+WACC)^y
  NL=(C+R_nonel+OM+DA+SV)/WACC, where WACC for year y = (1+WACC)^y.
  C and SV are zero for all years except the first and last of the project lifetime respectively.
- *2nd loop*: sum over all components.
  EL_tot=sum over components of EL
  NL_tot=sum over components of NL

Finally, the cost of electricity is evaluated and returned to RAVEN by
HYB_ELcost_tot = -NL_tot/EL_tot

In addition to the cost of electricity, the fuel consumption and the CO2 production are evaluated. As described above, these quantities are evaluated by a sum over all components

HYB_fuel_tot=[*component name*]_fuel*([*component name*]_productionEL*8760.0*
[*component name*]_capacity+ [*component name*]_productionBY*8760.0)*
[*component name*]_capacity/Efficiency)

HYB_CO2_tot=[*component name*]_CO2*([*component name*]_productionEL*8760.0*
[*component name*]_capacity+ [*component name*]_productionBY*8760.0)*
[*component name*]_capacity/Efficiency)

5.4 Future Work on the Cash Flow Model

The cash flow model so far implemented is fairly flexible, complete both from the aspect of manuals and input error handling and it is therefore approaching being a complete component of the modeling and simulation strategy for hybrid systems. As already mentioned it could be a reasonable extension to allow some of the parameters to be provided, both, as global and subsystem-based parameters. The code can accept an unbounded number of subsystems and it looks capable to accommodate all the needed information for the components tested so far. It is not to be excluded that the input structure will need to accommodate more information in the future.

6. EXAMPLE OF INTEGRATED SYNTHETIC TIME HISTORIES, CASH FLOW ANALYSIS, AND OPTIMISATION

6.1 Data Flow for Example Calculation

To test the new developed models and RAVEN capabilities in an integrated way, an example has been calculated. The goal of this example is to demonstrate the correct data flow between the RAVEN external models, in particular:

- The cash flow model
- The "synthetic time series"
- The simple system model (later to be replaced with Modelica)
- The optimization

As mentioned, the goal of the hybrid modeling and simulation work is to analyze the economy of a hybrid nuclear reactor on the electrical grid, where all the components on the grid are modeled dynamically and explicitly with the Modelica code. For the purpose of the demonstration of the RAVEN model and capability developments, Modelica has not been used, but a simple python model has been invoked to generate the utilization fractions for the different components. This python model is taken from [37] and referred to as "Simple system model".

The example considered for the prove of functionality of the RAVEN developments include four components:

- Hybrid nuclear reactor
- Gas turbine
- Wind farm
- Battery storage

The data flow of the example is shown in Figure 16. First, RAVEN samples the mean demand and the installed capacities for all components except the reactor, which is supposed to have a fixed capacity of 300MW electric. RAVEN then passes one sample of the mean demand and the installed wind capacity to the "Synthetic time series generation" model. This model generates time series for the demand and available wind capacity. Values of the demand and the available wind for every hour of a year are passed back to RAVEN. RAVEN passes these data together with the sampled installed capacities for all components to the "Simple system model". This external model computes the electricity produced, fuel consumed and CO_2 produced for each component in order to satisfy the demand curve. The next section gives a short description of this model. The production, fuel consumption and CO_2 production are then passed back to RAVEN. RAVEN passes these data together with the sampled installed capacities for all components to the cash flow model. This model computes and passes back to RAVEN the cost of electricity for this configuration of the system. The cash flow model is described in detail the section above.

It is worth mentioning that in this methodology, the optimization does not know exactly the demand of the future, even though the optimization is done over the whole year. Since a new synthetic time history is generated for each sample passed by RAVEN, the optimization can be considered for a given mean demand only, not for a given (known) demand history.

The combination of the three models and the passing of information between them are done using the newly developed RAVEN capability "EnsembleModel". The cash flow analysis model needs some additional inputs that are not provided directly by RAVEN. These inputs are stored in an external XML input file that is read by the RAVEN capability "ReadMoreXML" which was specifically developed to give external RAVEN models the possibility to read additional XML inputs.

Once the cost of electricity is passed back to RAVEN, RAVEN can perform the analysis of the data, such as statistical analysis or executing a limit surface search. Finally, the newly developed RAVEN optimization model (without constraints) presented above has been applied to find the system configuration that gives the minimum cost of electricity for a given demand.

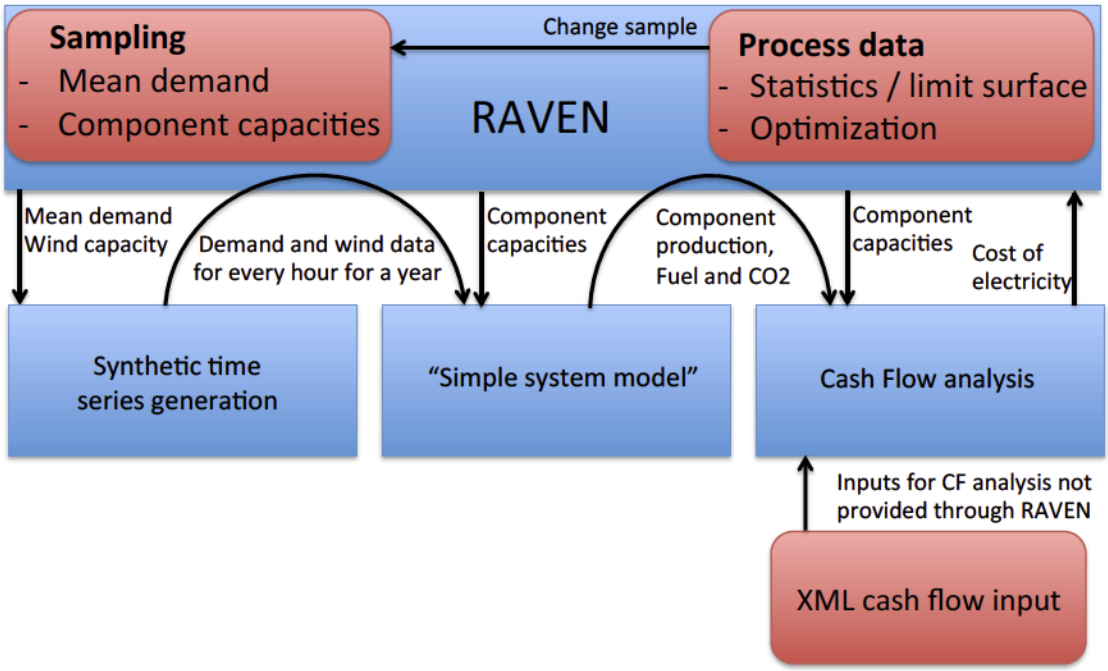


Figure 16. Data flow for example calculation

6.2 Simple System Model

As mentioned, in order to test new developments like the cash flow model or the "EnsembleModel" capability in RAVEN, an example has been calculated. The final goal is to use Modelica to model the components of the system in detail. For testing the new developments in RAVEN, using a complex Modelica model that integrates multiple components is not convenient. Therefore, for the test example, Modelica does not model the system, but a simple python system model taken from [37]. This model provides the amount of electricity produced during a year (utilization) for each component for a given demand history, wind (renewable) availability history and capacity for each component.

For each hour of the demand history, the configuration specified by the RAVEN input parameters is used to supply electricity. Each included component is dispatched based on its marginal cost of generation from least expensive to the most expensive. The exception is the nuclear reactor, which produces secondary product rather than putting electricity on the grid unless the other energy sources fail to cover demand in a given hour. It is assumed that selling heat is more lucrative than selling electricity for the nuclear reactor. The price of heat is taken into account in the cost model.

As shown in Figure 17, renewable generation is dispatched first because of its assumed zero marginal cost of generation. If excess wind energy is available it is then used to charge the integrated battery. If the demand is not covered by wind, electricity stored in the battery is dispatched second, then electricity from natural gas and the nuclear reactor. This dispatch order is based on the assumption that energy from the nuclear reactor is more economically used to produce potable water (secondary product) than to produce

electricity. The nuclear energy used for electricity generation to meet grid demand and the energy used for secondary commodity production are stored in separate variables. In this manner the revenue from the secondary product can be calculated later. Once the demand is covered or all resources have been dispatched, the utilization of each component is calculated using Equation 33, and a flag variable is set to indicate whether the hour was successfully covered by the given configuration. The utilization factor for each component and the number of "lost hours" (demand not satisfied) during the year is then passed back to RAVEN [37].

$$\text{Utilization} = \text{Energy Used}/(\text{Total Possible Generation}) \tag{33}$$

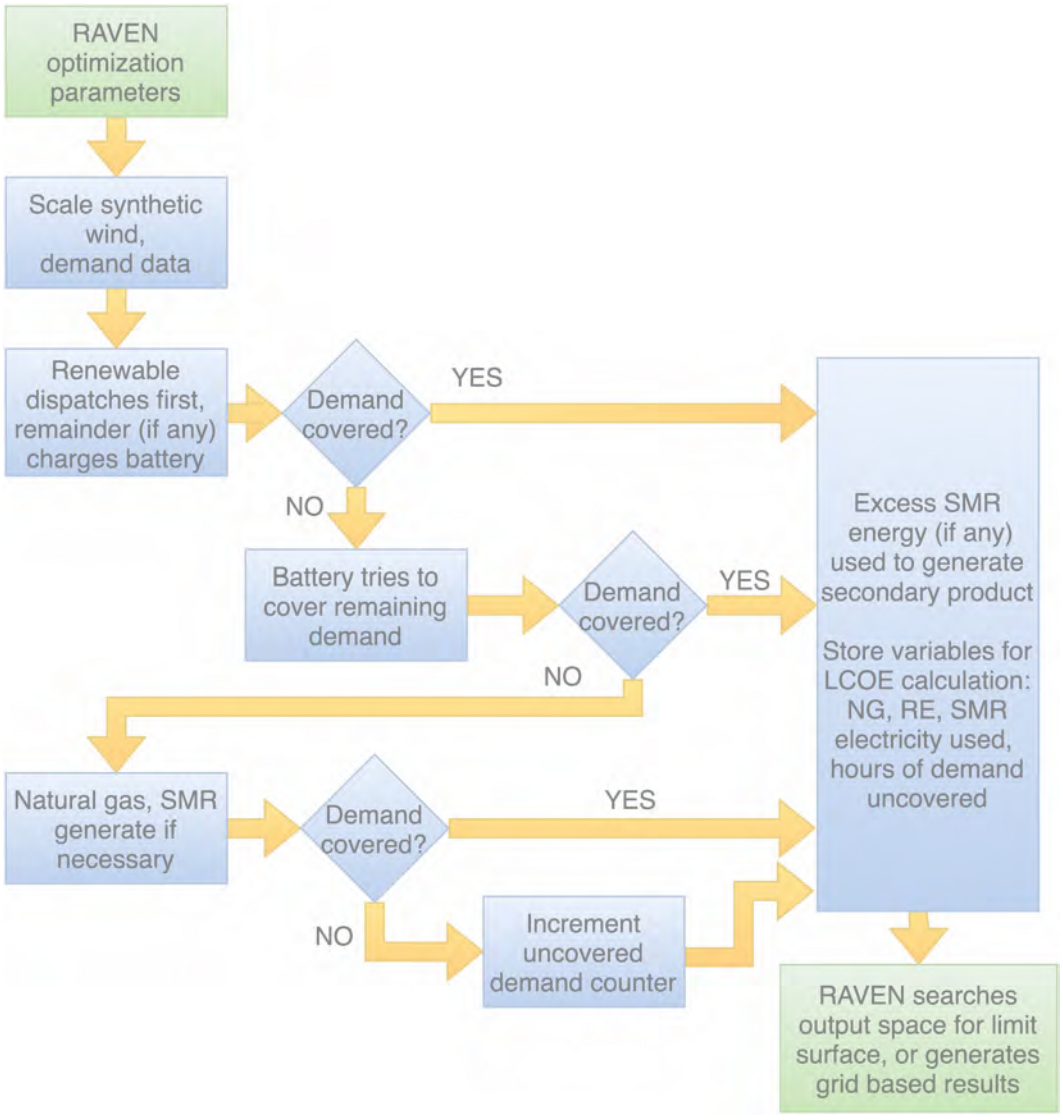


Figure 17. Flow Diagram of Python Cost Model [37]

6.3 Results

Is should be noted that the results shown here are solely to demonstrate the functionality of the RAVEN "EnsembleModel" capability to pass data between models as well as the functionality of the synthetic time history and the cash flow models. They use preliminary input data and the results do not necessarily allow an interpretation of the economics of the hybrid system.

As described in the data flow section above, first, a sample of mean demand and renewable capacity are passed to the synthetic time history generator. The generator passed back to RAVEN a time realization for both of them. The resolution is hourly and the length of realizations is one year. While a more detailed statistical analysis and comparison of the synthetic data to the measurements is presented in Section 3, just one synthetic realization for demand and wind availability are shown here in Figure 18 and Figure 19. In the shows example, RAVEN asked the synthetic time series generator to produce histories with an average of 10,000kW for both, mean demand and mean renewable capacity. As one can see, the time series comply with the mean demand and wind capacity passed from RAVEN. These figures are just an example for one realization of the demand and wind availability. Other averages of demand and wind can be asked by RAVEN depending on the sampling strategy chosen.

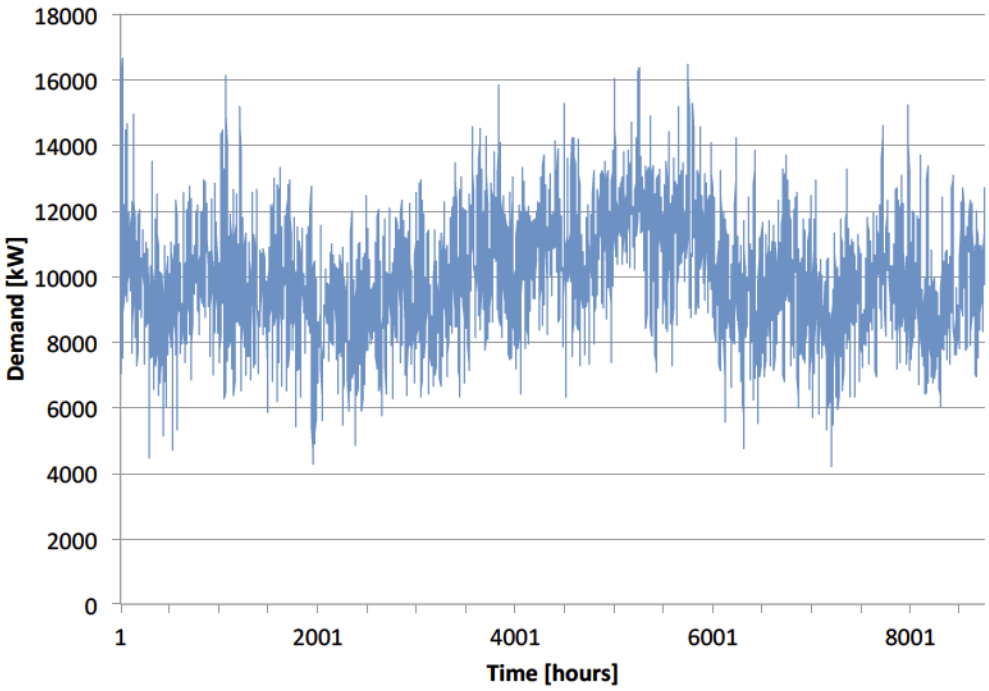


Figure 18. Synthetic time history realization for 10MW mean demand

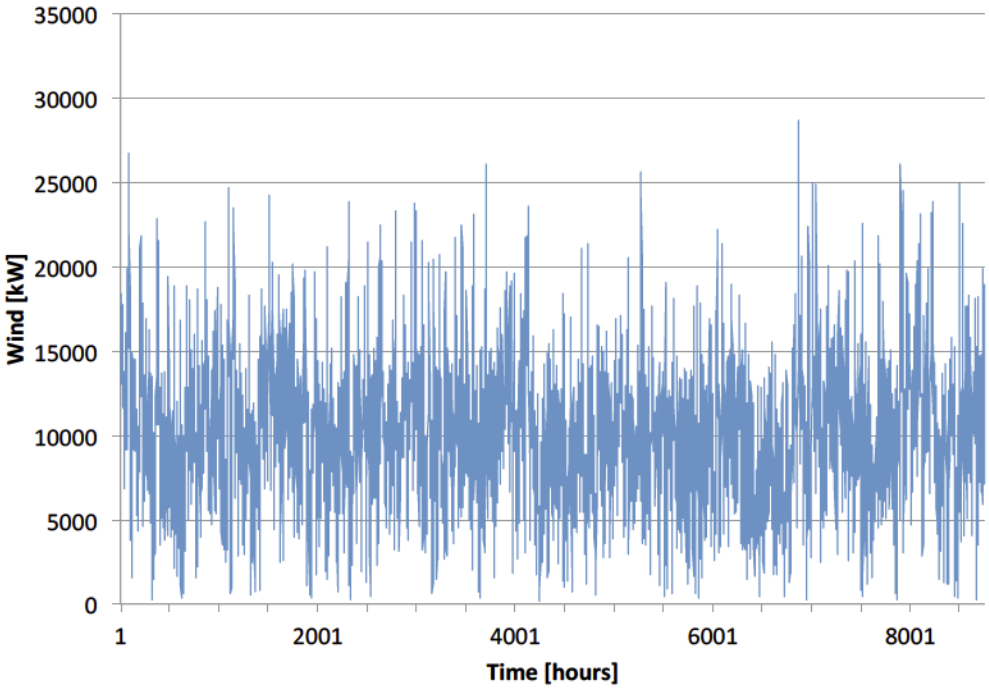


Figure 19. Synthetic time history realization for 10MW renewable capacity

Once the synthetic time histories are generated for a given sample set (Renewable, Gas turbine and Battery capacity and mean demand), the "simple system model" generates the utilization factors and the cash flow model computes the price of electricity for this system configuration and mean demand. Table 4 shows the input used for the cash flow analysis. It is worth noticing that the numbers used for this demonstration are purely demonstrative and not representative of a possible system configuration.

Table 4. Cash Flow Model Input for Model Functionality Demonstration

```
<Cash_Flow verbosity="1"> <!-- "0" all debug output, "1" some output, "100" only errors  -->

  <Global>
    <Project_life>30</Project_life> <!-- years -->
    <WACC>0.08</WACC> <!-- %/100 -->
    <tax>0.392</tax> <!-- %/100 -->
    <inflation>0.02</inflation> <!-- %/100 -->
  </Global>


  <Component name="Reactor">
    <Amort_time>60</Amort_time>  <!-- years -->
    <Efficiency>0.35</Efficiency>  <!-- to compute capacity(th)=capacity(el)/efficiency -->
    <Capital_per_capacity>3317.8</Capital_per_capacity> <!-- $/kW(el) -->
    <No_elec_revenue_price>0.025</No_elec_revenue_price>  <!-- $/[kW(th)*h] -->
    <OM_per_capacity>90.02</OM_per_capacity>  <!-- $/kW(el) -->
    <OM_per_production>0.00049</OM_per_production>  <!-- $/kW(el+th)h -->
```

```
      <DepreciationAmortisation>0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07
                        0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07
                        0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07
                        0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07
                        0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07
                        0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07</DepreciationAmortisation>
      <Salvage_value>0.0</Salvage_value>  <!-- $ -->
    </Component>


    <Component name="GasTurbine">
      <Amort_time>30</Amort_time>  <!-- years -->
      <Efficiency>0.6</Efficiency>  <!-- to compute capacity(th)=capacity(el)/efficiency -->
      <Capital_per_capacity>947.5</Capital_per_capacity> <!-- $/kW(el) -->
      <No_elec_revenue_price>0.0</No_elec_revenue_price>  <!-- $/[kW(th)*h] -->
      <OM_per_capacity>11.7</OM_per_capacity>  <!-- $/kW(el) -->
      <OM_per_production>0.002</OM_per_production>  <!-- $/kW(el+th)h -->
      <DepreciationAmortisation>0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07
                        0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07
                        0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07</DepreciationAmortisation>
      <Salvage_value>0.0</Salvage_value>  <!-- $ -->
    </Component>


    <Component name="Renewable">
      <Amort_time>20</Amort_time>  <!-- years -->
      <Efficiency>1.0</Efficiency>  <!-- to compute capacity(th)=capacity(el)/efficiency -->
      <Capital_per_capacity>1922.99</Capital_per_capacity> <!-- $/kW(el) -->
      <No_elec_revenue_price>0.0</No_elec_revenue_price>  <!-- $/[kW(th)*h] -->
      <OM_per_capacity>30.3</OM_per_capacity>  <!-- $/kW(el) -->
      <OM_per_production>0.0</OM_per_production>  <!-- $/kW(el+th)h -->
      <DepreciationAmortisation>0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07
                        0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07</DepreciationAmortisation>
      <Salvage_value>0.0</Salvage_value>  <!-- $ -->
    </Component>


    <Component name="Battery">
      <Amort_time>15</Amort_time>  <!-- years -->
      <Efficiency>1.0</Efficiency>  <!-- to compute capacity(th)=capacity(el)/efficiency -->
      <Capital_per_capacity>1964.0</Capital_per_capacity> <!-- $/kW(el) -->
      <No_elec_revenue_price>0.0</No_elec_revenue_price>  <!-- $/[kW(th)*h] -->
      <OM_per_capacity>51.0</OM_per_capacity>  <!-- $/kW(el) -->
      <OM_per_production>0.06</OM_per_production>  <!-- $/kW(el+th)h -->
      <DepreciationAmortisation>0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07
                        0.07 0.07 0.07 0.07 0.07</DepreciationAmortisation>
      <Salvage_value>0.0</Salvage_value>  <!-- $ -->
    </Component>
  </Cash_Flow>
```

RAVEN has been asked to sample the four variables mean demand and renewable, gas turbine and battery storage capacity on a grid from 100WM to 1,000MW while the reactor capacity is constant 300MW. Ten equally spaced sample points have been asked for each variable, i.e. a 4 dimensional grid with 10,000 points total. Figure 20 shows the cost of electricity (color of points) for the three dimensions mean demand, renewable and gas turbine capacity. The data has been post-processed before plotting to include only points that meet the demand (less than 30 hours lost per year). Although the goal is to show functionality of the model and the preliminary inputs used, one can see that demand is covered for small mean demands and bigger mean demands as the capacities grow (to be noticed that reactor capacity is kept constant). This is expected. Furthermore, one can see that the cost of electricity goes down as the mean demand goes up. This is also expected, since when demand and capacity match well, capital cost is distributed over a bigger production for bigger demand, which leads to a better economy of scale.

As a second example, RAVEN has been asked to find the limit surface for meeting demand itself. The grid is the same as in the previous example. Figure 21 shows the limit surface found by RAVEN. Black points indicate that demand has not been met while white points indicate that the demand has been met for this configuration of the system. One can see that the limit surface found by RAVEN (without evaluating the whole grid) corresponds to the one found by manually post processing the full grid data as shown in Figure 20.

The whole set of calculation was run in parallel using 24 cores on a single computational node. The parallel dispatching is currently managed by the RAVEN infrastructure and the scalability detected was almost ideal.

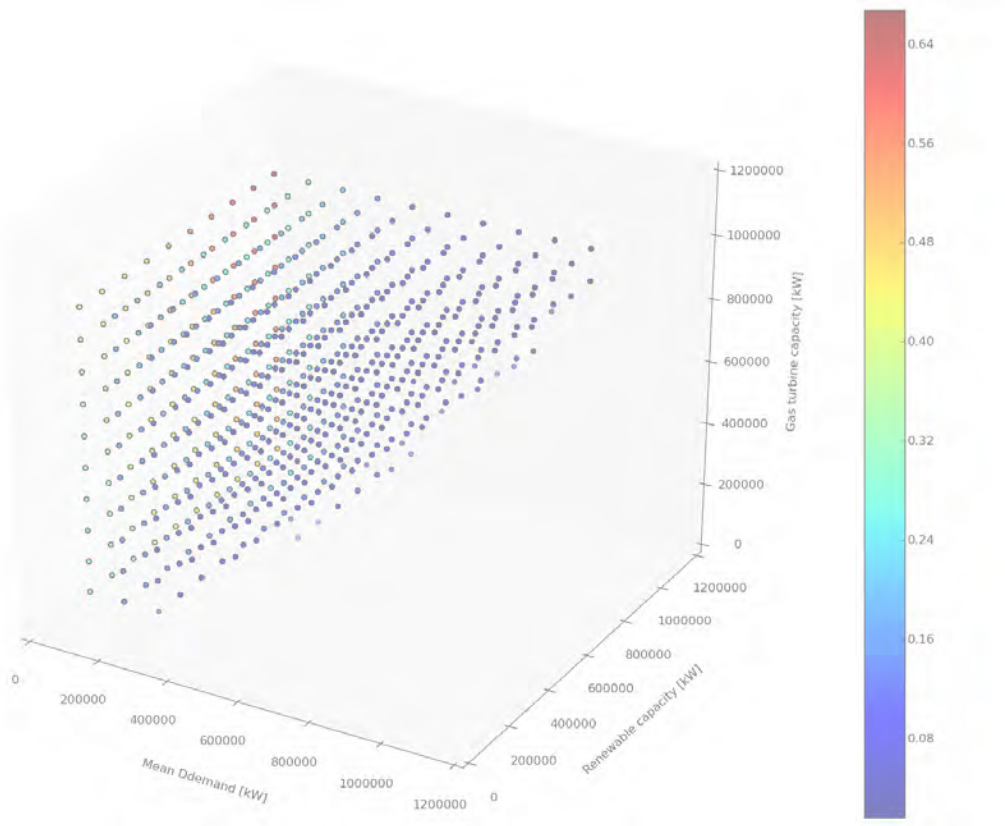


Figure 20. Cost of electricity: RAVEN grid evaluation

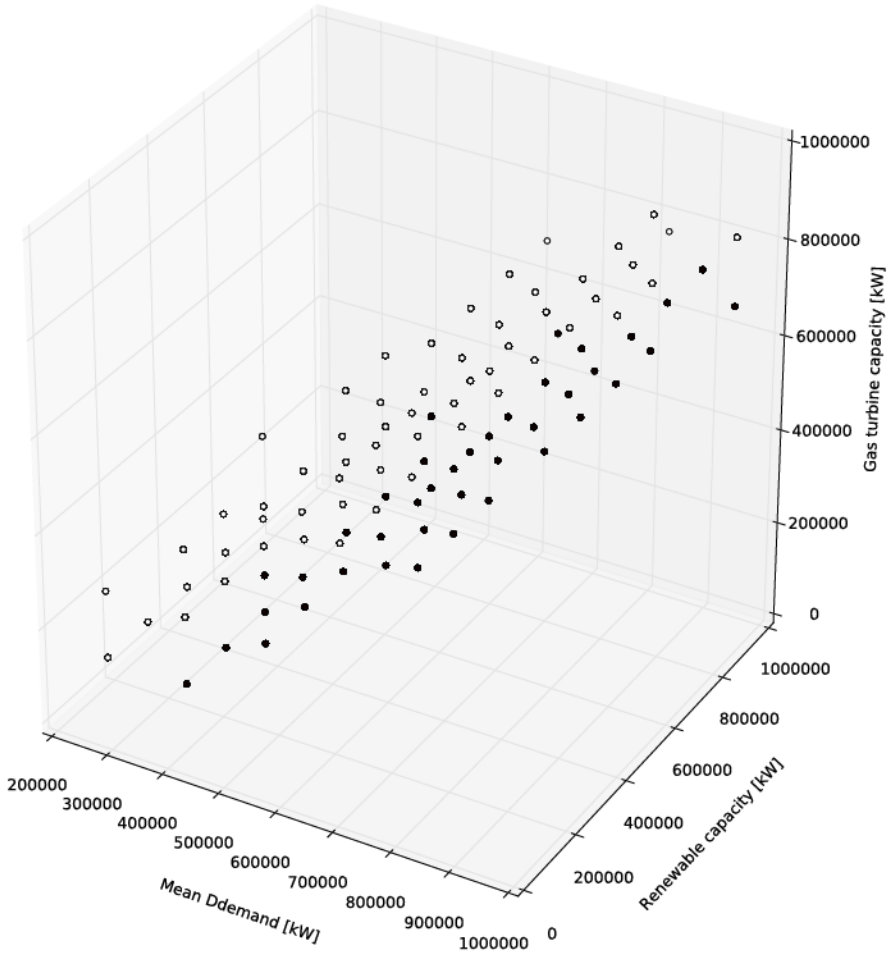


Figure 21. Limit surface for meeting demand (less than 30 hours lost per year). Black dots are failures; white dots signify successes

As a last example, RAVEN has been asked to optimize the cost of electricity. Since work on the constraint optimizer is still ongoing, an unconstraint optimization has been performed, only the minimum and maximum of the input variables has been bound. The same variables as for the grid and limit surface evaluation has been used, i.e. the optimizer can vary the mean demand and the renewable, gas turbine and battery storage capacities between 100MW and 1,000MW while the reactor capacity is kept at 300MW. The optimizer has computed 250 iterations of the optimization using a carefully chosen set of optimizer parameters. Figure 22 shows the optimization path for the three dimensions mean demand, renewable and gas turbine capacity while Figure 23 indicates how these dimensions converge. It can be seen that the algorithm converges to a minimum cost of electricity of $0.016. Again, these numbers are just to illustrate the proper working of the integrated example with the optimizer and do not reflect a real minimum cost of electricity for a NHES.

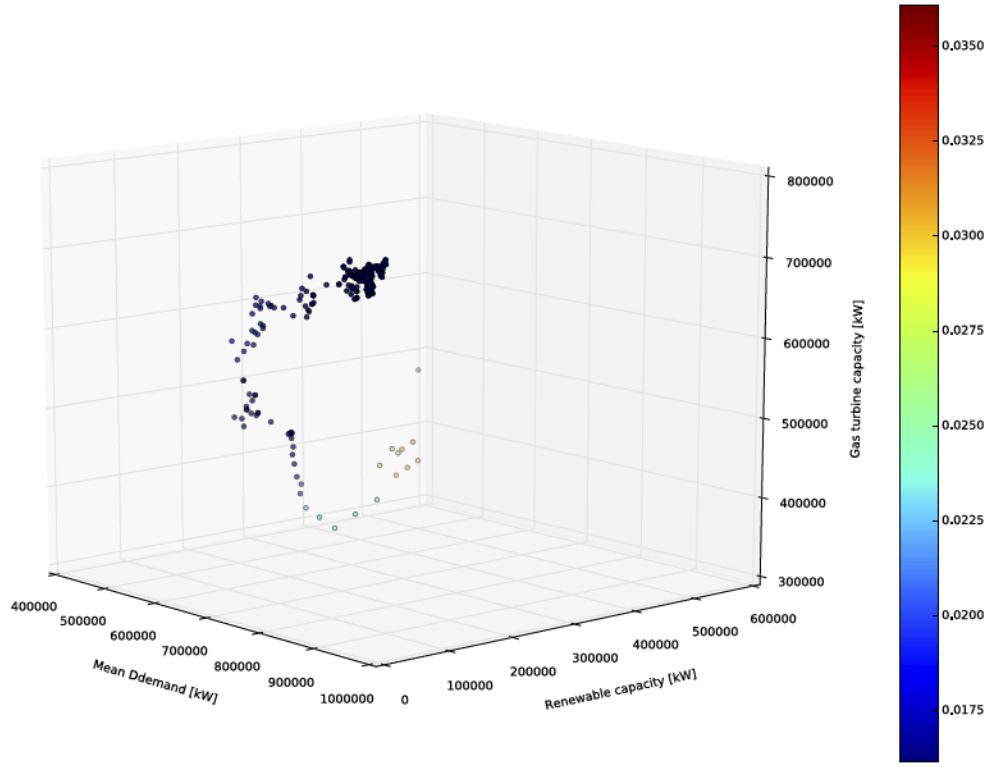


Figure 22. Unconstraint optimization for cost of electricity: optimizer path

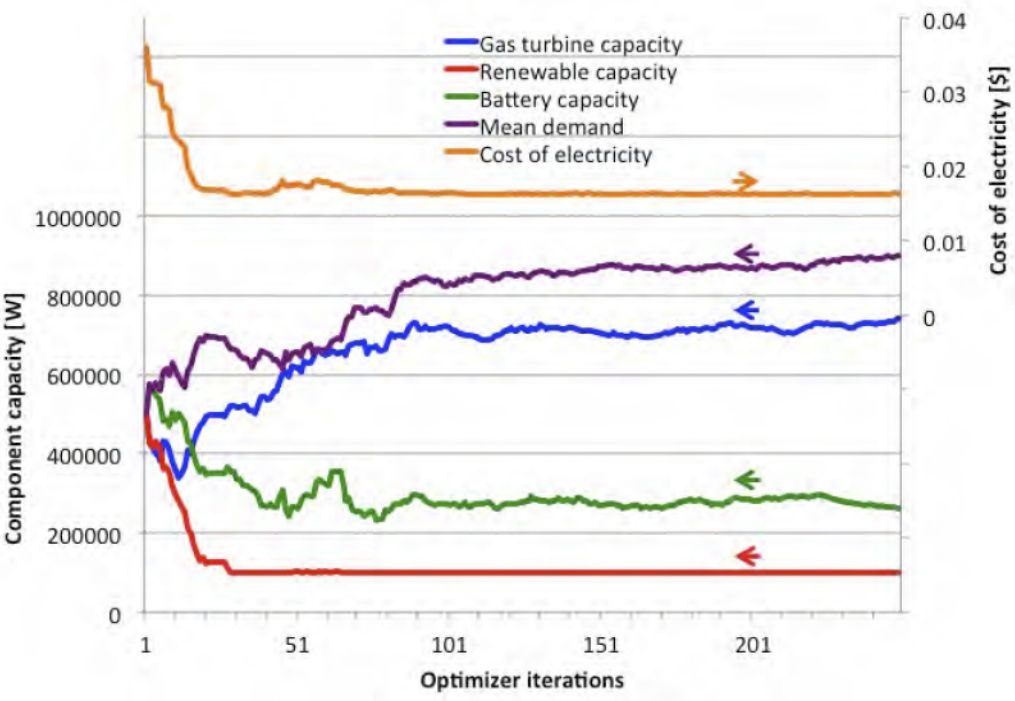


Figure 23. Iteration of optimization variables and cost of electricity

7. FUTURE ACTIVITIES

The activities performed in FY16 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 (with respect to the development of the framework/driver):

- Acquire an unlimited processor license for Dymola for the Linux environment
- Complete the optimization driver, in particular:
  - Extend the set of optimization algorithms available.
  - Complete the constraints handling in current implementation to allow RAVEN to solve more effectively constrained optimization problem.
- 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 performance and the value of the economic analysis framework proposed.

8. CONCLUSIONS

A sizable quantity of groundwork has been completed during FY16. The derivation, implementation and integration with RAVEN of the synthetic time series allows to define the boundary condition in which the NHES operate by creating stochastically self similar time histories. The unlimited number of time histories, which is possible to generate using this approach, is the first cornerstone that allows approaching the stochastic optimization problem. The algorithm chosen, Fourier analysis to remove trends, and ARMA to account for the aleatory component, has proven to provide excellent results. Both synthetic wind speed and grid load time histories demonstrated good statistic conformance towards historical database.

Moreover, a new type of sampler, namely “optimizer”, has been developed in RAVEN with highly object-oriented design. In particular, simultaneous perturbation stochastic approximation algorithm has been implemented. The optimizer is shown to be capable of driving the model to optimize a scalar objective function without constraint in the input space. In order to handle the constraints, several possible approaches have been proposed and discussed, and are currently under construction to improve the optimization capability in RAVEN for solving constrained optimization problem. With respect last year now the optimization is fully integrated in RAVEN and any sub-sequential development will be naturally carried on within the raven framework using the proper QA.

The “external model” API present in raven has been used to develop a cash flow analysis tool. At the moment the implemented external model is capable to accept an unlimited number of subsystem constituting one of many possible NHES configuration. The cash flow analysis return the cost of electricity that zeros the NPV, to RAVEN which can use it just as simple output or as optimization variable.

The cash flow model is of course dependent on the physical performance of the NHES (e.g. electricity production). While in the future the physical representation of the system will be supplied as an external code by written in Modelica language for testing purpose a simple model has been build to represent a possible NHES (also interfaced with RAVEN using the external model API).

The coordination, by RAVEN, of the creation and feeding of the synthetic time histories to the external model representing the NHES, and the creation of the corresponding cash flow has been tested.

The linking between the different components of the simulation has been managed by using the ensemble model feature of the RAVEN package. This implementation has been proven to work not only in single processor but also in parallel on several cores. At this moment it was not necessary go beyond 24 cores since the problem was so simple that most of the time was spent in I/O.

At this point the RAVEN supporting infrastructure is almost ready to be linked with the NHES representation by the middle of FY 17.

9. REFERENCES

1. C. Rabiti, 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. H. Elmqvist, et al., "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. http://www.3ds.com/products-services/catia/products/dymola
4. C. Rabiti, A. Alfonsi, J, Cogliati, D. Mandelli, R. Kinoshita, S. Sen, C. Wang, J. Chen, RAVEN user manual, INL/EXT-15-34123, Revision 5, February 2016.
5. H. Liu, H.-Q. Tian, C. Chen, Y.-f. Li, A hybrid statistical method to predict wind speed and wind power, Renewable Energy 35 (8) (2010) 1857-1861.
6. F. Cassola, M. Burlando, Wind speed and wind energy forecast through Kalman filtering of Numerical Weather Prediction model output, Applied Energy 99 (2012) 154-166.
7. P. Louka, G. Galanis, N. Siebert, G. Kariniotakis, P. Katsafados, Pytharoulis, G. Kallos, Improvements in wind speed forecasts for wind power prediction purposes using Kalman filtering, Journal of Wind Engineering and Industrial Aerodynamics 96 (12) (2008) 2348-2362.
8. F. O. Hocaoglu, M. Fidan, O. N. Gerek, Mycielski approach for wind speed prediction, Energy Conversion and Management 50 (6) (2009) 1436-1443.
9. T. El-Fouly, E. El-Saadany, M. Salama, Grey predictor for wind energy conversion systems output power prediction, IEEE Trans. Power Syst. 3 (21) (2006) 1450-1452.
10. L. Suganthi, A. A. Samuel, Energy models for demand forecasting A review, Renewable and sustainable energy reviews 16 (2) (2012) 1223-1240.
11. P. Meibom, R. Barth, B. Hasche, H. Brand, C. Weber, M. O'Malley, Stochastic optimization model to study the operational impacts of high wind penetrations in Ireland, IEEE Trans. Power Syst. 26 (3) (2011) 1367-1379.
12. J. M. Morales, R. Minguez, A. J. Conejo, A methodology to generate statistically dependent wind speed scenarios, Applied Energy 87 (3) (2010) 843-855.
13. A. Papavasiliou, S. S. Oren, R. P. O'Neill, Reserve requirements for wind power integration: A scenario-based stochastic programming framework, IEEE Trans. Power Syst. 26 (4) (2011) 2197-2206.
14. X.-Y. Ma, Y.-Z. Sun, H.-L. Fang, Scenario generation of wind power based on statistical uncertainty and variability, IEEE Trans. Sustainable Energy 4 (4) (2013) 894-904.
15. J. W. Taylor, P. E. McSharry, R. Buizza, Wind power density forecasting using ensemble predictions and time series models, IEEE Trans. Energy Conversion 24 (3) (2009) 775-782.
16. H. Mori, E. Kurata, Application of gaussian process to wind speed forecasting for wind power generation, in: Proc. 2008 IEEE Int. Conf. Sustainable Energy Techn., IEEE, 956-959, 2008.
17. D. Lee, R. Baldick, Short-term wind power ensemble prediction based on Gaussian processes and neural networks, IEEE Trans. Smart Grid 5 (1) (2014) 501-510.
18. D. Lee, R. Baldick, Synthesis of sample paths of wind power through factor analysis & cluster analysis, in: Proc. 2013 North American Power Symposium (NAPS), Manhattan, KS, 1-6, 2013.
19. D. Lee, R. Baldick, Future wind power scenario synthesis through power spectral density analysis, IEEE Trans. Smart Grid 5 (1) (2014) 490-500.
20. L. J. Soares, M. C. Medeiros, Modeling and forecasting short-term electricity load: A comparison of methods with an application to Brazilian data, International Journal of Forecasting 24 (4) (2008) 630-644.

21. K. K. Sumer, O. Goktas, A. Hepsag, The application of seasonal latent variable in forecasting electricity demand as an alternative method, Energy policy 37 (4) (2009) 1317-1322.
22. E. Gonzalez-Romera, M. A. Jaramillo-Moran, D. Carmona-Fernandez, Monthly electric energy demand forecasting based on trend extraction, IEEE Trans. Power Syst. 21 (4) (2006) 1946-1953.
23. H. S. Hippert, C. E. Pedreira, R. C. Souza, Neural networks for short-term load forecasting: A review and evaluation, IEEE Trans. Power Syst. 16 (1) (2001) 44-55.
24. Z. Wang, S. Bian, Y. Liu, Z. Liu, The load characteristics classification and synthesis of substations in large area power grid, International Journal of Electrical Power & Energy Systems 48 (2013) 71-82.
25. N. Amjady, F. Keynia, Short-term load forecasting of power systems by combination of wavelet transform and neuroevolutionary algorithm, Energy 34 (1) (2009) 46-57.
26. N. Steckler, A. Florita, J. Zhang, B. Hodge, Analysis and Synthesis of Load Forecasting Data for Renewable Integration Studies, in: Proc. 12th International Workshop on Large-Scale Integration of Wind Power into Power Systems, London, England, Oct. 22-24, 2013.
27. G. E. Box, G. M. Jenkins, G. C. Reinsel, G. M. Ljung, Time series analysis: forecasting and control, John Wiley & Sons, 2015.
28. S. Wilcox, W. Marion, Users manual for TMY3 data sets, National Renewable Energy Laboratory Golden, CO, 2008.
29. J. M. Finkelstein, R. E. Schafer, Improved goodness-of-fit tests, Biometrika 58 (3) (1971) 641-645.
30. C. Spall, Implementation of the simultaneous perturbation algorithm for stochastic optimization, IEEE Transactions on aerospace and electronic systems, vol. 34, no. 3, pp. 817–823, 1998.
31. J. C. Spall, An overview of the simultaneous perturbation method for efficient optimization, Johns Hopkins apl technical digest 19.4 (1998): 482-492.
32. J. L. Maryak, and D. C. Chin, Efficient global optimization using SPSA, American Control Conference, 1999. Proceedings of the 1999. Vol. 2. IEEE, 1999.
33. C. Rabiti, R. S. Cherry, W. R. Deason, P. Sabharwall, S. M. Bragg-Sitton, R. D. Boardman, Framework for the Economic Analysis of Hybrid Systems Based on Exergy Consumption, INL/EXT-14-23934, August 2014.
34. L. G. Eldenburg, and S. K. Wolcott, Cost Management, Hoboken, NJ: John Wiley & Sons, Inc, 2005.
35. R. C. Higgins, Analysis for Financial Management, eleventh edition, Mcgraw Hill Higher Education, 2016.
36. http://people.stern.nyu.edu/adamodar/New_Home_Page/datafile/wacc.htm
37. T. Baker, Analysis of Nuclear Hybrid Systems with Battery Storage using Levelized Cost of Electricity, Master Thesis, Idaho State University, 2016.