

Deployment and Overview of RAVEN Capabilities for a Probabilistic Risk Assessment Demo for a PWR Station Blackout

Cristian Rabiti
Andrea Alfonsi
Diego Mandelli
Joshua Cogliati
Richard Martineau
Curtis Smith

June 2013

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



Deployment and Overview of RAVEN Capabilities for a Probabilistic Risk Assessment Demo for a PWR Station Blackout

**Cristian Rabiti
Andrea Alfonsi
Diego Mandelli
Joshua Cogliati
Richard Martineau
Curtis Smith**

June 2013

**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**

DISCLAIMER

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

CONTENTS

1.	Introduction	3
1.1	RAVEN: a tool to increase the risk management capabilities for nuclear power plants	3
1.2	RAVEN: a Tool to Ensure the Deployment of the RISMIC Concept.....	4
1.3	RAVEN: as an Overall Control System.....	4
1.4	Overview of the Document	4
2.	Software Infrastructure Overview	5
2.1	Mathematical Formulation of the Problem	5
2.1.1	System and Control Logic.....	5
2.1.2	Modeling of Probabilistic Behaviors	6
2.1.3	The Risk Weighted Formulation.....	7
2.2	Software Infrastructure for RELAP-7 Interaction.....	8
2.3	GUI Software Infrastructure	9
2.4	Artificial Intelligence Aided Discovery Framework.....	10
2.5	Dynamic Event Tree approach.....	12
3.	Reference Plant Analysis.....	13
3.1	Description of the graphical input process.....	15
3.2	Control logic implementation	19
3.3	21	
3.4	Online Monitoring.....	21
3.4.1	Postprocessors.....	22
3.4.2	Visualize	22
3.5	Result Analysis	23
4.	Introducing Statistical Behaviors.....	25
4.1	PWR SBO Test Case.....	25
4.2	Modeling.....	27
4.3	Results.....	28
4.3.1	Explanation of the Effect on the Maximum Clad Temperature Using a Random Distribution on the Failure Temperature	32
5.	Conclusion.....	33
5.1	References.....	34
6.	APPENDIXES.....	36
6.1	APPENDIX A: Station Black Out Inputs	36
6.1.1	Plant and Raven Tools Input.....	36
6.1.2	Control Logic Input.....	1

1. Introduction

1.1 RAVEN: a tool to increase the risk management capabilities for nuclear power plants

Starting in 2012, Idaho National Laboratory (INL) has been engaged to provide tools and methodologies to support a new approach to safety assessment and management of safety related issues for Nuclear Power Plants (NPPs).

This activity is placed under the Risk Informed Safety Margin Characterization (RISMC) [2] umbrella that is one of the technical paths of the Department Of Energy (DOE) Light Water Reactor Sustainability (LWRS) Program. RISMC takes a broader approach to risk evaluation by generating more comprehensive information. This information is used to gain a more accurate estimation of risk margins and to highlight feasible improvements that seek the optimal point between risk mitigation and economical viability of the nuclear energy option. This is, of course, done without taking any judgmental approach with respect to risk, which is a regulatory body task, and the economical viability, which is up to the single states and the open electricity market. RISMC is therefore composed of methodologies and tools. The augmented information that RISMC seeks to generate, as enabling decision tool, is not holistic with respect all possible metrics and operational aspects of a power plant; it is rather channeled toward the improvement of safety and the associated economical impact. While these are indeed two very broad metrics, they still allow a focus within the development of the underlying tools that, in a resource-constrained environment (DOE founding, computational, mathematical) maximize the benefit of the program to the nuclear industry.

Part of the enabling information comes from an increased accuracy in the evaluation of risk margins and their associated uncertainties. New codes are, currently, under development to provide the needed increase in fidelity, and the one in the most advanced status of development is RELAP-7 [3]. RELAP-7 is currently under development at INL and is foreseen to become the next of the RELAP series (currently INL is the licenser of RELAP5-3D). RELAP-7, as its predecessors, is a nuclear system safety analysis code that will embed new numerical schemes to increase the accuracy of results, allow the user to analyze the full plant life time, and span coherently a larger situation range (e.g. same equation set from shock waves to slow transients). One of the major strengths of RELAP-7, not yet mentioned, is the MOOSE [4] underlying platform. This approach offers, not only all the benefits of a modern modular development approach (ease of maintenance and quality insurance), but also leverages the high fidelity set of codes developed, or under development, in this environment. The common environment easily allows RELAP-7 to interact with one of the other MOOSE based applications such as Bison (fuel performance) [5], Grizzly (material aging for pressure vessel)[6], RattleSnake (neutronics)[7].

While all of it addresses the need of higher fidelity, it does not provide information on the impact of uncertainties arising from lack of modeling capabilities (closure laws are still necessary) and physical knowledge (uncertainty in experimental data impairs an exact knowledge of physical parameters), and the intrinsic probabilistic nature of some events that might impact nuclear power plant safety (pump failure, earthquakes, etc.).

Insofar as the uncertainties in our risk knowledge, arising from intrinsically probabilistic events, have been addressed by Probabilistic Risk Analysis (PRA), the uncertainties in our models and physical analysis have been accounted by margins imposition.

One of the innovative approaches suggested by RISMC is the simultaneous analysis of these two contributions in a probabilistic sense. This improves the capability to assess higher fidelity margins with respect to situations that might bear unwanted consequences.

This is one of the main challenges that has been assigned to the RAVEN project. RAVEN combines uncertainty quantification (UQ) with probabilistic analysis of risk (PRA). To avoid

being overwhelmed by either of the two tasks, RAVEN uses the risk and economical impact focus to narrow the span of information that needs to be generated.

1.2 RAVEN: a Tool to Ensure the Deployment of the RISMC Concept

It is expected that, as with any new technology, RISMC will face an initial period of resistance and a strong momentum will be needed to overcome the tipping point after which its tools and concepts will be broadly adopted in the nuclear community.

A coherent strategy needs to be employed in order to mitigate the resistance as much as possible, and, thus, the friendliness of new tools and clearness of information should be part of this strategy. In this respect, RAVEN has also been assigned the task of being an information manager: a discriminator of what is relevant or not to properly evaluate risk.

RAVEN is the interface between the user and the whole underlying set of new tools, focusing their work toward the production and delivering of the relevant information.

This is the part of the project for which a Graphical User Interface (GUI) has been created allowing the generation of RELAP-7 input (in the future it will be extended to more MOOSE based applications) and analyzing the simulation results.

Visual inspection of results is a fundamental aspect of engineering analysis, and at the same time RAVEN is developing an environment where modern data mining techniques will be also accessible to the users. Those techniques will be used to filter the information and carefully discover the sources of risk in terms of critical components, leading physical phenomena, or uncertainty sources. This information is of course twofold: the identification of risk sources can, in fact, be used to find possible mitigation strategies.

1.3 RAVEN: as an Overall Control System

As for all PRA software, the capability to control the plant evolution during the simulation is a plus for uncertainty propagation. For these reasons, a strict interaction between RELAP-7 and RAVEN is a key of the long-term success of the overall project. In system safety analysis codes, a similar need is expressed by the implementation of the control logic of the plant. As a consequence, the optimization of resources imposes the integration of this task under a common project that is RAVEN. Consequently, the plant control logic is simulated by RAVEN; this also offers the flexibility to easily implement proprietary control logic without changing RELAP-7 source code. This feature is also a factor for the quick deployment of RELAP-7 to the industry.

In summary, from a user prospective, RAVEN is a tool that:

- Easily generates complex plant layout including modeling information and control logic implementation (RELAP-7 GUI, and Control Logic)
- Allows following the simulation via visual interaction with the code while running
- Determines the calculation flow to achieve the most accurate evaluation of risk accounting for probabilistic behavior and uncertainty propagation
- Visualizes simulations results (thousands and more) and provides the data mining capability to deeply understand the plant behavior
- Provides the capability to investigate risk mitigation strategies by suggesting directions and quickly assessing impacts

1.4 Overview of the Document

The scope of this document is to illustrate how the actual RAVEN capabilities can be applied to a Station Blackout (SBO) analysis for a simplified PWR model.

The following sections will illustrate:

- A general overview of the mathematical formulation and software infrastructure
- The input generation (comprehensive of operational control logic) for the RELAP-7 code via the GUI and the online visualization capabilities
- A qualitative analysis of the simulation results
- The introduction of probabilistic aspect in the simulation (PRA and UQ) and description of the set up for Monte Carlo analysis.
- An analysis of the results, with particular emphasis on a comparative approach to highlight the effects of uncertainties on the final outcome
- Conclusions with overview of the ongoing development and its contribution to the RAVEN project

2. Software Infrastructure Overview

In the following paragraphs, a quick overview of the underlining mathematical formulation and the software infrastructure supporting the analysis described later on will be provided to help the reader getting familiar with the terminology and the tools.

2.1 Mathematical Formulation of the Problem

2.1.1 System and Control Logic

A nuclear power plant is a complex system but its mathematical representation will be always reducible to a first order non-linear differential system of equations. The system (Eq. 1) represents the time evolution of the variables $\bar{\theta}$ fully describing the plant system (phase space).

$$\frac{\partial \bar{\theta}(t)}{\partial t} = \bar{\mathcal{H}}(\bar{\theta}(t), t)$$

Eq. 1

Without going into the details, the system above entails a continuous behavior of the velocity (in the Hamiltonian sense) $\bar{\mathcal{H}}$ of the system in the phase space. This is not obviously always true, and therefore the derivative should be eventually interpreted in the distributional sense. From now on, this argument will not be further analyzed and it will be assumed that differential operators are defined in the proper space, so that notation is consistent.

The first step is to perform a partition of the phase space so that \bar{x} are the variables solved by RELAP-7 (e.g. pressure, temperature) and \bar{c} the ones directly controlled by the set of equation representing the plant control logic (e.g. valve opening/closing).

$$\begin{cases} \bar{\theta} = \begin{pmatrix} \bar{x} \\ \bar{c} \end{pmatrix} \\ \frac{\partial \bar{x}}{\partial t} = \bar{F}(\bar{x}, \bar{c}, t) \\ \frac{\partial \bar{c}}{\partial t} = \bar{C}(\bar{x}, \bar{c}, t) \end{cases}$$

Eq. 2

Since the control logic reacts on changes happening only in a limited subset of the phase space, this could be reflected in its mathematical representation by the introduction of a subset of the \bar{x} space called ‘monitored’ space \bar{m} . \bar{m} is extracted from the whole phase space by a projection operator $\bar{M}(\bar{x}, t)$. The final system is:

$$\begin{cases} \frac{\partial \bar{x}}{\partial t} = \bar{F}(\bar{x}, \bar{c}, t) \\ \bar{m} = \bar{M}(\bar{x}, t) \\ \frac{\partial \bar{c}}{\partial t} = \bar{C}(\bar{m}, \bar{c}, t) \end{cases}$$

Eq. 3

Given the discontinuous nature of the control logic, when it comes to choosing the numerical integration scheme, to avoid instability, infinite loops, and other problems, it is best to decide to use an operator split approach. The corresponding time discretization scheme is illustrated in the following:

$$\begin{cases} \frac{\partial \bar{x}}{\partial t} = \bar{F}(\bar{x}, \bar{c}_{t_{i-1}}, t) \\ \bar{m} = \bar{M}(\bar{x}, t) \\ \frac{\partial \bar{c}}{\partial t} = \bar{C}(\bar{m}, \bar{c}_{t_{i-1}}, t) \end{cases} \quad t_{i-1} \leq t \leq t_i = t_{i-1} + \Delta t_i$$

Eq. 4

2.1.2 Modeling of Probabilistic Behaviors

As already mentioned, RAVEN will be used to perform PRA and therefore a probabilistic formulation of the outcome is necessary. In reality RAVEN extends the PRA analysis by including also Uncertainty Quantification. Without going into the details of the derivation, the final equation [8, 9] solved by RAVEN is:

$$\begin{aligned} \frac{\partial \Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0)}{\partial t} &= - \sum_i \frac{\partial}{\partial \theta_i^c} (A_i(\bar{\theta}^c, \bar{\theta}^d, t) \Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0)) \\ &+ \frac{1}{2} \sum_{i,j} \frac{\partial^2}{\partial \theta_i^c \partial \theta_j^c} (B_{i,j}(\bar{\theta}^c, \bar{\theta}^d, t) \Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0)) \\ &+ \int (W(\bar{\theta}^c | \bar{\theta}'^c, \bar{\theta}^d, t) \Pi(\bar{\theta}'^c, t | \bar{\theta}_0^c, t_0) - W(\bar{\theta}'^c | \bar{\theta}^c, \bar{\theta}^d, t) \Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0)) d\bar{\theta}'^c \\ \frac{\partial \Pi(\bar{\theta}^d, t | \bar{\theta}_0^d, t_0)}{\partial t} &= \sum_i W(\bar{\theta}^d | \bar{\theta}_i^d, \bar{\theta}^c, t) \Pi(\bar{\theta}_i^d, t | \bar{\theta}_0^d, t_0) - W(\bar{\theta}_i^d | \bar{\theta}^d, \bar{\theta}^c, t) \Pi(\bar{\theta}^d, t | \bar{\theta}_0^d, t_0) \end{aligned}$$

Eq. 5

Where:

- $\bar{\theta}^c$: partition of the phase space containing all the continuous variables
- $\bar{\theta}^d$: partition of the phase space containing all the discontinuous variables
- $\Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0)$: probability of the plant system to be found in $\bar{\theta}^c$ given as initial condition $\bar{\theta}_0^c$
- $\Pi(\bar{\theta}^d, t | \bar{\theta}_0^d, t_0)$: probability of the plant system to be found in $\bar{\theta}^d$ given as initial condition $\bar{\theta}_0^d$
- A_i : is the speed of the system in the phase space along the coordinate i

- $B_{i,j}$: operator that generates a diffusive behavior of the system location probability, due to possible internal stochastic behavior of the system like cracks propagation, plant noises etc.
- $W(\bar{\theta}^c | \bar{\theta}'^c, \bar{\theta}^d, t)$: is the probability of transition, by time unit and distance unit in the phase space, from $\bar{\theta}'^c, \bar{\theta}^d$, to $\bar{\theta}^c$

Figure 1 provides a graphical interpretation of the difference between a deterministic and a probabilistic system behavior. In the first case the system location evolves along a trajectory line; while in the second case, even starting from an exact point, the system spreads its presence probability over the phase space while time passes.

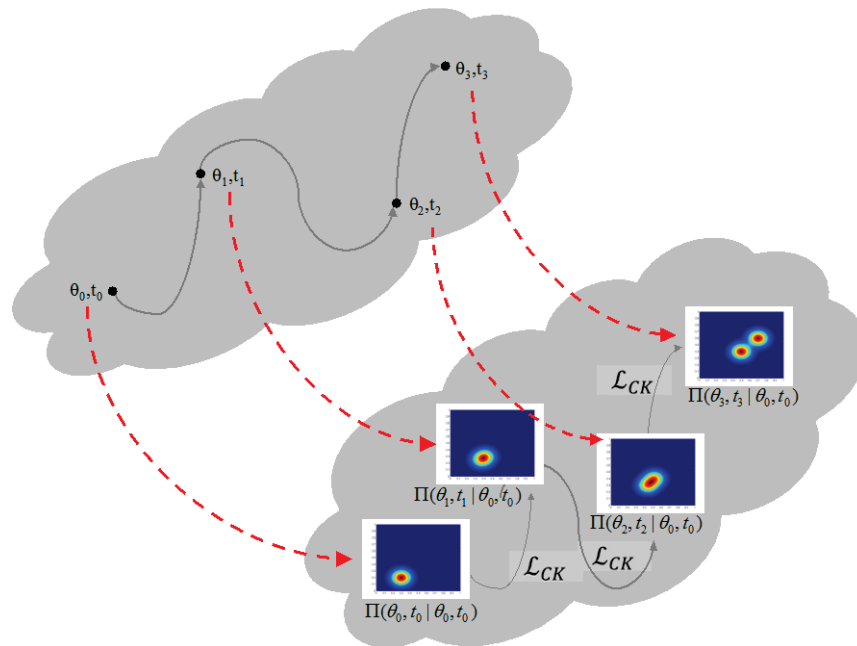


Figure 1: Deterministic (left) vs. probabilistic (right) system evolution

At the moment the solution of the equation Eq. 5 is performed in RAVEN through Monte Carlo, Dynamic Event Tree (currently under testing), and soon also by Stochastic Polynomials. To be noticed that aside few cases (presence of Wigner like processes [8, 9]), the Monte Carlo sampling could be reduced to sampling of initial conditions and/or equation parameters [9].

2.1.3 The Risk Weighted Formulation

Eq. 5 is in literature referenced as Chapman Kolmogorov. This equation is very difficult to solve, especially when the number of dimensions of the phase space is very large, as in the case of interest for the RAVEN project, where probabilistic behavior is analyzed in conjunction with uncertainty propagation.

Luckily not all the information contained in the solution of Eq. 5 has the same value. In fact, it should not be forgotten that the goal of this project is the risk evaluation.

$$Risk = Probability \times Consequences$$

Probability is provided by the solution of Eq. 5 and the consequence metrics of interest may be suggested by stakeholders such as the regulatory body, plant management, and, at the end, by the software user.

A very simple example of how this observation could be used to limit the space for which it is necessary to know the solution of Eq. 5 is the following:

$$Consequence == \begin{cases} 1 & \text{if clad temperature exceeds max allowable temperature} \\ 0 & \text{if clad temperature does not exceed max allowable temperature} \end{cases} \quad \text{Eq. 7}$$

In such a case the risk evaluation reduces to the probability of the system to have a set of initial condition such as, at a certain point in time, the consequence function is equal to 1. This defines a very precise zone of the input space delimited by what is call *limit surface*. Figure 2 graphically illustrates the concept.

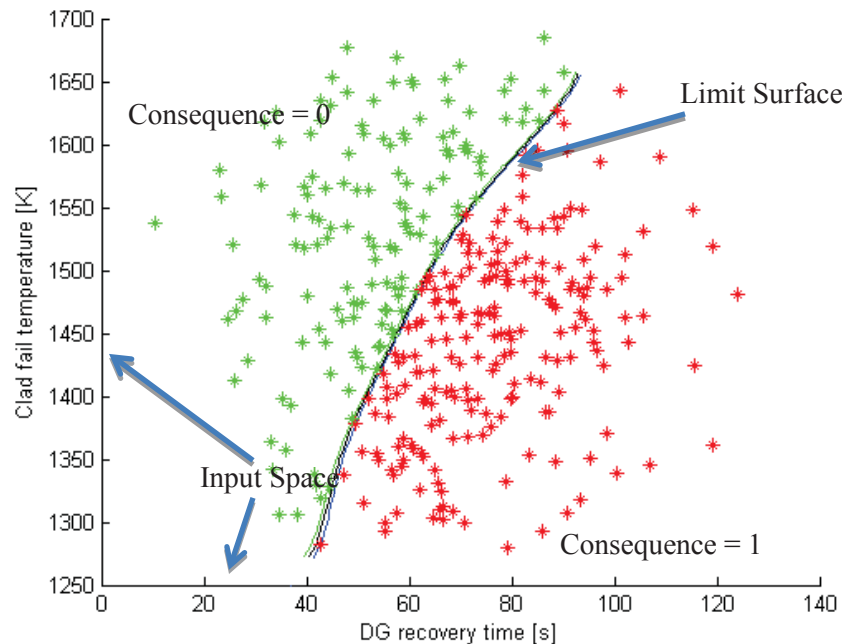


Figure 2: Limit surface

This concept will be discussed in more detail later on in a practical application for the PWR SBO analysis.

2.2 Software Infrastructure for RELAP-7 Interaction

This section provides an overview of how the above-described set of equations is implemented in the MOOSE framework. The MOOSE framework is a software environment that provides tools for the solution and the handling of PDE (Partial Differential Equations) based problems. MOOSE uses LibMesh [10] to translate the equations into their numerical forms and than PETSc [11] to solve the problem. Around these two fundamental capabilities, MOOSE provides several handling features to structure the problem and to perform post processing. One of the most relevant capabilities of this framework is that it allows the user to quickly implement the numerical solution of new sets of equations and coupling physics.

RELAP-7 is one of the MOOSE based applications, and it contains the set of equations to be solved and then the numerical form is solved by MOOSE.

RAVEN plays the role of plant control logic. From a software point of view this capability is implemented in the following way (nomenclature is referred from Eq. 3):

1. Via the input files RAVEN recognizes:
 - a. Which variables \bar{c} the user wants to control
 - b. Which variables \bar{m} (and the associated projection operator $\bar{M}(\bar{x}, t)$) the control logic will use to make decisions
 - c. The set of probability distributions that will be used during the simulation to sample the input space
2. RAVEN inquires to RELAP-7 to know the location of the information it needs to retrieve and control
3. This information is used at each time step to directly interact with MOOSE

Figure 3 illustrates this communication pattern.

RAVEN does not contain software to specifically represent any control logic but opens a channel between the simulation (MOOSE) and a Python interface where the user can implement his own control logic laws. In the Python interface all the controlled and monitored variables are made accessible to users, as well as few utility functions (probability distribution function and premade control functions).

The MOOSE project not only provides an easy implementation and leveraging software opportunity but also enforces a strict Quality Assurance (QA) protocol for all the applications built within its environment.

All software developed on the MOOSE framework is currently under sub-version control with an extensive regression test suite. New software developments are collected in a shared repository, automatically tested, and code coverage (percent of the code controlled by the regression tests) is constantly monitored.

2.3 GUI Software Infrastructure

The mechanics allowing the implementation of the GUI is a good example of how it is possible to take advantage of the common MOOSE platform. The MOOSE environment is capable to provide the information on what are the needed parameters for each component.

In the same way the variables that could be controlled (e.g. pump head, component failure status), monitored (e.g. temperature, pressure) and the respective projection operators (e.g. average, maximum) are registered inside the code itself. *Peacock* is a generic GUI for MOOSE based applications that is capable of automatically retrieving such information. The structure of the information allows *Peacock* to automatically construct entry tabs.

This makes input file creation easier for someone unfamiliar with the keywords and file structure. *Peacock* is also extensible. RAVEN implements additional features in *Peacock* that are described below.

A graphical visualization of the nuclear power plant layout is displayed as it is created. The user, through the GUI, can visualize the results while the simulation is still running. Two types of information are available: a three dimensional projection of the solution \bar{x} generated by RELAP-7 on the plant layout, and the 2D plots of each monitored or controlled quantity (respectively \bar{m} and \bar{c}). The solution field is communicated from RELAP-7 to the GUI via a *.vtk* file type and the monitored and controlled via a *.csv* (Comma Separated Values) file type. Both files are continuously updated at each time step, to allow continuous monitoring.

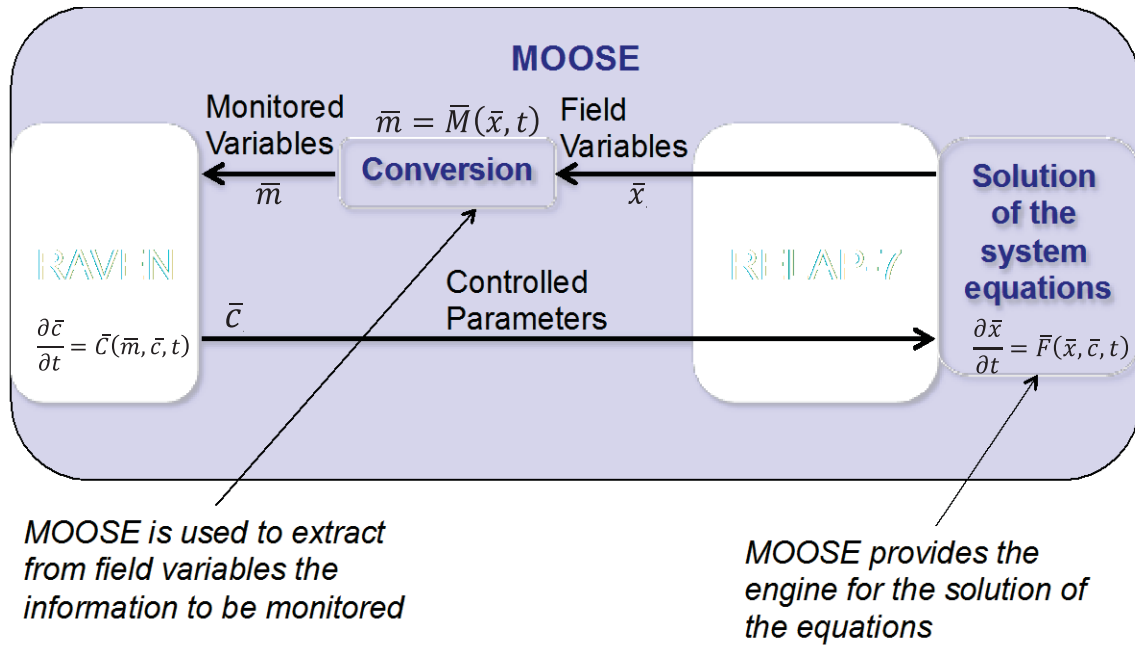


Figure 3: Software implementation

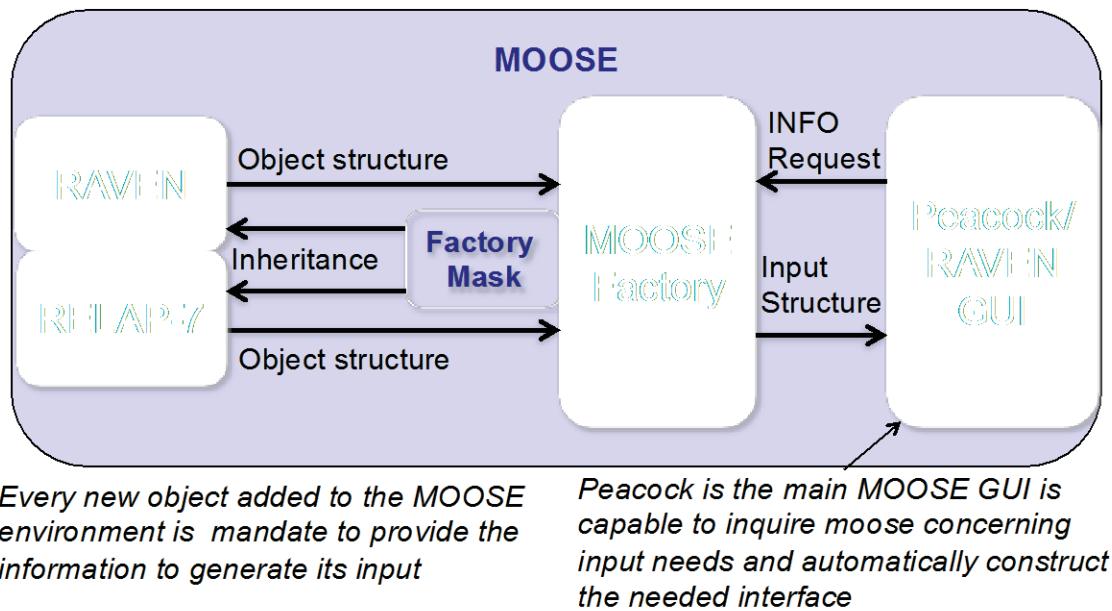


Figure 4: Automatic process to construct the GUI interface

2.4 Artificial Intelligence Aided Discovery Framework

The last component of RAVEN is an external wrapper that is used to perform sampling of the simulation for different value of the input parameters.

The scheme used for this sampling is shown in Figure 5. While this is the general approach the implementation of the DET based analysis has a slightly different approach that is, for this reason, described in a separate paragraph.

In general, the simulation sequence could be synthesized by the following steps:

1. Few points in the input space are selected and a RELAP-7/RAVEN simulation is performed in correspondence of those entries
2. The set of results are used to train a Reduced Order Model (ROM): a mathematical approximation of the plant system specific to the case under evaluation
3. The ROM is inquired to evaluate the risk function
4. The regions with low accuracy (not well represented by the initial set of points in the input space) and highly important toward an accurate evaluation of the risk function, are then chosen to be further investigated
5. More evaluation (RELAP-7/RAVEN simulations) are performed in such regions
6. The ROM is enriched by the new information acquired and either the process restart from point 3 or it stops being achieved the needed accuracy

To provide a practical example it is helpful to refer to the consequence function described in the paragraph on ‘the Risk Weighted Formulation.’ In that case a suitable choice for the reduced order model could be a classifier (a ROM returning a Boolean value) that will be False if the maximum clad temperature is exceeded during the simulation and True otherwise. This classifier is, in the case illustrated in this report, built using Support Vector Machines (SVM).

The sequence is:

1. Initial Monte Carlo sampling of the input space: a low number of simulations is run for a set of value of the input parameters randomly generated
2. The initial set of simulations is used to train the SVM classifier: the ROM
3. The ROM is used to predict the location of the limit surface in the input space
4. Additional sample points are chosen along the limit surface
5. The simulation on those points are used to refine the ROM
6. If the limit surface has changed position then return to step 3 otherwise stop the process (i.e. convergence is reached)

This approach has the great advantage to adaptively focus on the regions of the input space that are important to proper evaluate the risk function, being therefore more effective. The development of ROMs algorithms is long and expensive; therefore a general interface to import the scikit-learn open source library has been generated [11].

Currently the iterative process is not yet in place so in the presentation of the SBO only the Monte Carlo results and the corresponding limit surface are shown.

This component of the RAVEN software has also the capability to spawn parallel runs of the underling code (RELAP-7/RAVEN) so to take advantage of large computer clusters to speed up the Monte Carlo process. For example during the simulation presented in this report ~120 cores were used to run 120 simulations at the time.

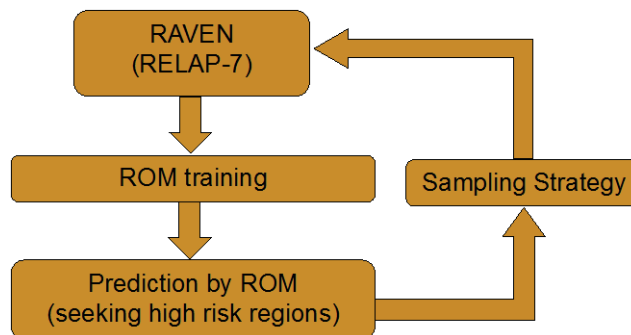


Figure 5: Usage of a ROM to perform guided sampling

2.5 Dynamic Event Tree approach

Conventional Event Tree (ET) based methodologies are extensively used as tools to perform reliability and safety assessment of complex and critical engineering systems. One of the disadvantages of these methods is that timing/sequencing of events and system dynamics is not explicitly accounted for in the analysis while it could be very important.

In order to overcome these limitations a “dynamic” approach is needed. The Dynamic Event Tree (DET) technique brings several advantages, among which the fact that it simulates system evolution in a way that is consistent with the progression of the accident scenario. In DET, event sequences are run simultaneously starting from a single initiating event. The simulation branches at user provided times and physical conditions. A couple of examples might clarify the methodology:

Branching at a certain point in time

- A valve is set to start a new branch every 20 minutes where it moves its status from available to failed
- Valve starts the simulation being available
- After the first 20 minutes of simulation a parallel branch is started where the valve is failed (branch 2), while in the root simulation (branch 1) the valve is still available
- For the next 20 minutes the valve will be available in the branch 1 while failed in branch 2. This might lead to a different evolution of branch 1 with respect branch 2
- After an additional 20 minutes the branch 1 will create another branch (3) and so on

Branching at a given physical conditions

- A valve has a certain probability to fail every time it gets used (failure on demand) by the control system
- Every time the control system demands the usage of the valve a new branch is generated where the valve is failed while the root branch move forward with the valve available

Situations could be much more complex, and RAVEN can handle almost all possible combinations like, for example, multiple outcomes (several branches started at the same time). The user has to set a time limit at which the branches will stop or an event which occurrence will also stop the simulation.

Each branch has associated a probability that represents the cumulative likelihood of the specific set of events leading the system to that branch. In this way it is always possible to associate to a specific outcome its own probability. Figure 6 provides a visual representation of a DET evolution.

The DET methodology has been developed in RAVEN and has been included in the RAVEN external Python manager (Artificial Intelligence Aided Discovery Framework).

Currently DET is already available in RAVEN but the application of this methodology to the PWR SBO has just started and is part of a later deliverable in September.

Figure 6: Dynamic Event Tree Scheme

3. Reference Plant Analysis

A simplified PWR model has been setup based on the parameters specified in the OECD main steam line break (MSLB) benchmark problem [13]. The reference design for the benchmark is derived from the reactor geometry and operational data of the TMI-1 Nuclear Power Plant (NPP), which is a 2772 MW two loop pressurized water reactor.

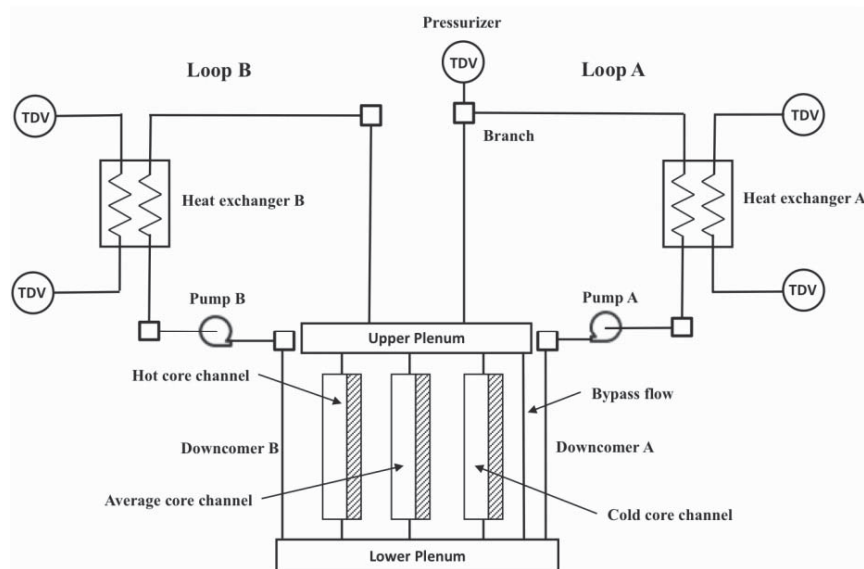


Figure 7: Scheme of the PWR model

Figure 7 shows the scheme of the PWR model. The reactor vessel model consists of the Downcomers, the Lower Plenum, the Reactor Core Model and the Upper Plenum. Three Core-Channels (components with a flow channel and an heating structure) were used to describe the reactor core. Each Core-Channel is representative of a region of the core (from one to thousands of real cooling channels and fuel rods). In this analysis, the core model consists of three parallel Core-Channels (hot, medium and cold) and one bypass flow channel. They represent the inner and hottest zone, the mid and the outer and colder zone of the core. The Lower Plenum and Upper Plenum are modeled with RELAP-7 Branch models. There are two primary loops in this model – Loop A and Loop B. Each loop consists of the Hot Leg, a Heat Exchanger and its secondary side

pipes, the Cold Leg and a primary Pump. A Pressurizer is attached to the Loop-A piping system to control the system pressure. Since a complex Pressurizer model has not been implemented yet in the current version of RELAP-7 code, a Time Dependent Volume (pressure boundary conditions) has been used instead.

To circumvent the lack of two phases flow of RELAP-7 code (now present, but not at the time of the simulation set up), single-phase counter-current heat exchanger models, with high mass flow rate, are implemented to mimic the function of steam generators (i.e. to transfer heat from the primary to the secondary).

Figure 8 shows the core layout of the PWR model. The core height is 3.6576 m. The reactor consists of 177 fuel assemblies subdivided in 3 zones. The 45 assemblies in zone 1 are represented by the hot core channel, the 60 assemblies in zone 2 and 72 assemblies in zone 3 are respectively represented by the average core channel and the cold core channel. The fuel assembly geometry data is taken from reference [13]. The reactor is assumed to be at end of cycle (EOC), 650 EFPD (24.58 GWd/MHMT average core exposure), with a boron concentration of 5 ppm, and Xe and Sm at the equilibrium. The 3-D core neutronics calculation results for the hot full power condition are presented in reference [13].

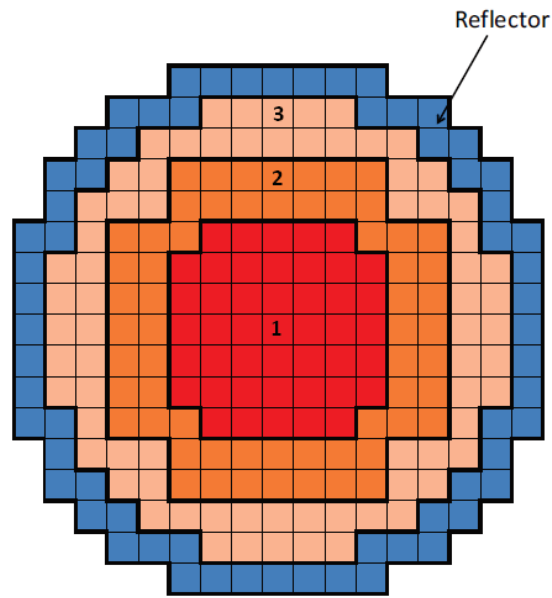


Figure 8: Core Zone Correspondence

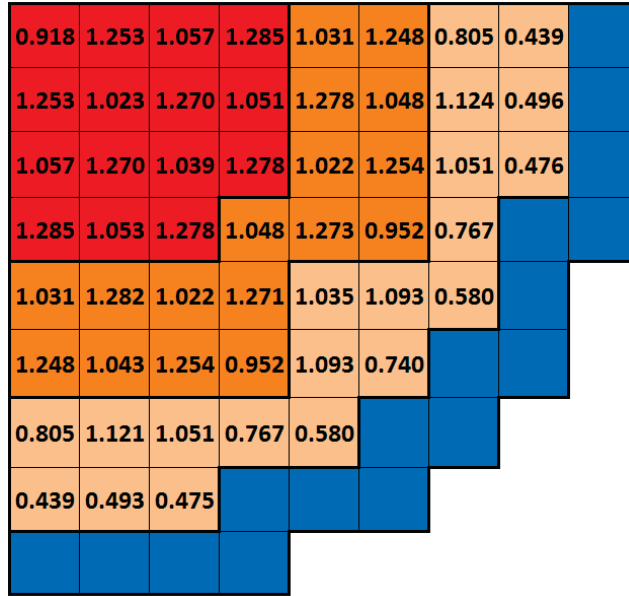


Figure 9: Assembly Relative Power

Figure 9 shows the relative assembly radial power distribution for a quarter of the core. Using the values presented in Figure 9, the power distribution fraction and power density for each Core-Channel is calculated and shown in the following table. The power density is used as input to RELAP-7 to calculate the heat source.

Table 1: Power distribution factor for representative channels and average pellet power

Core Channel	Power Distribution Factor	Average fuel pellet power density (W/m^3)
Hot	0.3337	3.90×10^8
Average	0.3699	3.24×10^8
Cold	0.2964	2.17×10^8

3.1 Description of the graphical input process

Input file creation begins by running Peacock from the directory containing the RELAP-7 code. An existing input file to begin with may also be specified. Peacock, after inquiring RAVEN for the proper input structure, initializes the GUI and eventually visualizes an already existing input file.

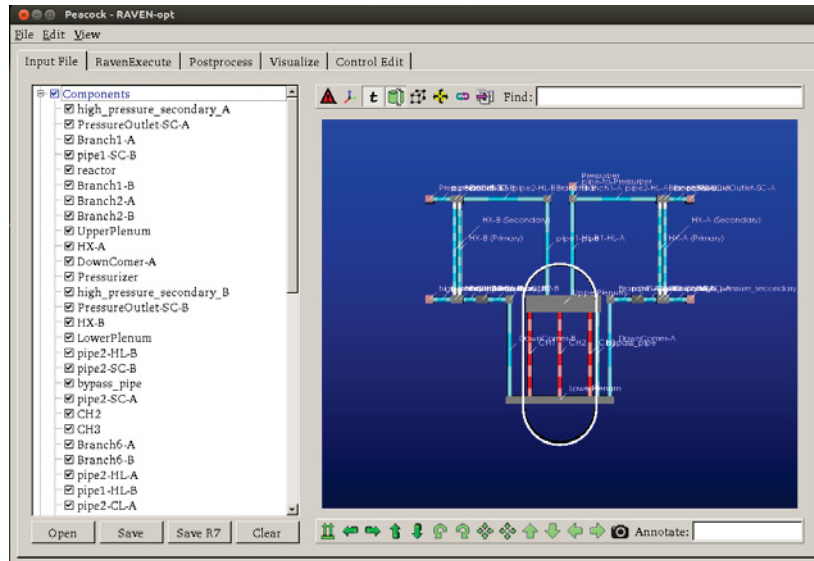


Figure 10: Peacock GUI for RAVEN

Figure 10 shows the Peacock GUI specific to RAVEN. It includes the visual representation of the plant as well as additional tabs: “RavenExecute” and “Control Edit”. On the left, the different sections of the input file are shown. The section containing the structure of the plant is expanded to a list of individual components. The checkboxes are used to quickly activate or deactivate particular parts of the input file.

On the right side, the plant layout is drawn. The user may navigate (pan, rotate, zoom) using either the mouse, or alternatively using buttons below the model display. Figure 11 shows a zoomed in view of the same model shown in Figure 10 with the navigation buttons highlighted. Also shown are the text labels for each component. If the mouse pointer is positioned over a component and left immobile (also known as “hovering”) the name and type are temporarily displayed in a box. In Figure 11, this is being shown for “DownComer-B” which is a pipe.

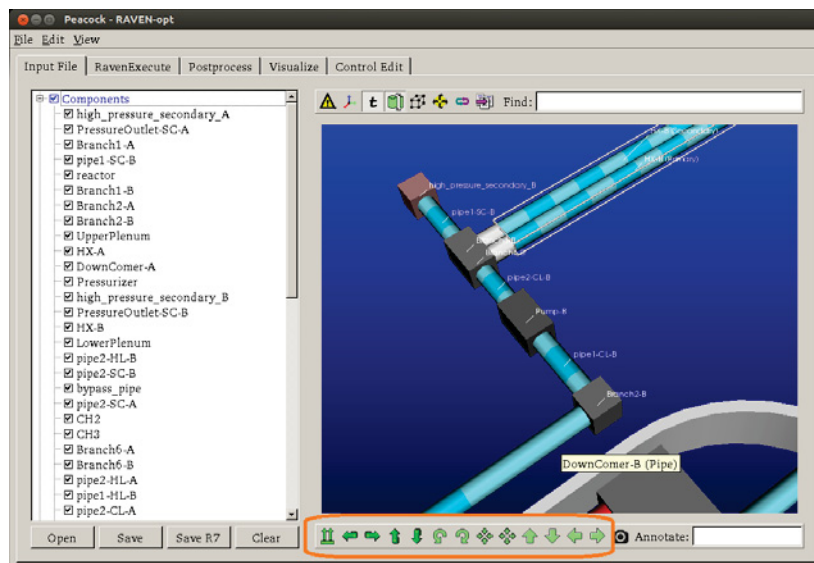




Figure 11: Model of Plant with Navigation Buttons Highlighted


There are also some helpful tools available at the top of the drawn model (Figure 12).





Figure 12: Upper Toolbar


 (Flashes red and yellow) The presence of this icon indicates that there are problems detected in the model. By clicking, it will display a selectable list of all of the problems listed by component.

 Toggles the display of labeled axes at the origin of the model. The default is off.


 Toggles the display of text indicating each the name of each component. The default is on.

 Toggles the display of all pipe objects between their actual physical size (based on cross-sectional area) and a uniform value. In many cases it is easier to comprehend the model using the constant. The default is uniform.

 Toggles the display of all components between solid (shown in Figure 11) and wire frame. Wire frame is useful to observe parts of the model not visible in solid mode. The default is solid.

 Toggles display of the current center of rotation for the model. In some cases (particularly when viewing large models) it may be useful to adjust the depth of the rotation center.

 Causes the model to be refreshed.

 Causes, if open, the “parameter edit” window (see below) to be brought to the front. This makes it easier to find it when covered by other windows.

Find: By typing into this box searches the list of component names. All names starting with the typed text are shown in a list. Selecting one causes it to flash.

The parameters for an individual item may be accessed by either double-clicking the list entry on the left or on the drawn component in the plant layout. If not already open, the “parameter edit” window is created and shown with a sub-window where the component parameters are editable. Several sub-windows can be simultaneously shown (see Figure 13).

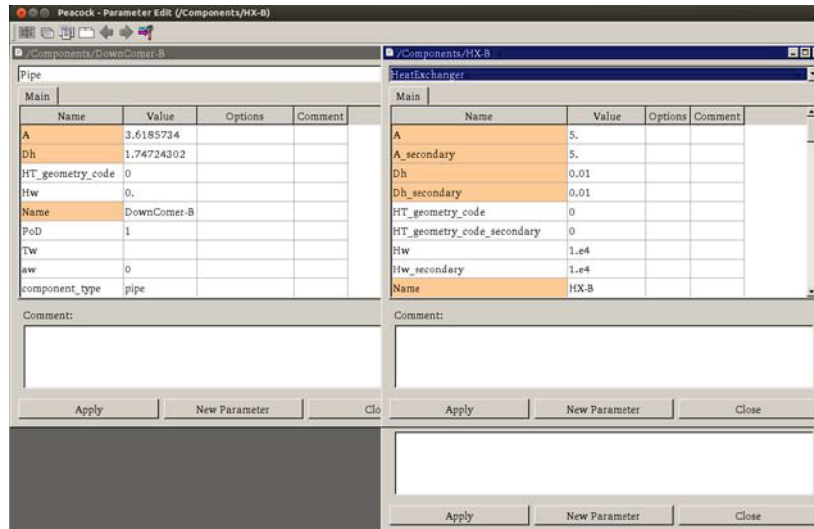


Figure 13: Parameter Window with Three Sub-Windows

Buttons are also provided to arrange the windows in various ways, as well as scroll through them one at a time. Any changes made to a component will take effect when its “Apply” button is pressed. Peacock will not allow changes to be applied until all mandatory values for a component have been entered. Mandatory entries are highlighted in orange as shown in Figure 13.

A context menu is available for any component by right-clicking on it. Figure 14 shows the context menu for “UpperPlenum”, which is of type “ErgBranch”. This menu will list all ports that may be connected to another component. Each of these provides a submenu allowing connections to be made and removed. In the example, UpperPlenum has four inputs: CH1, CH2, CH3, and bypass_pipe. When adding connections, only those actually eligible are listed for selection.

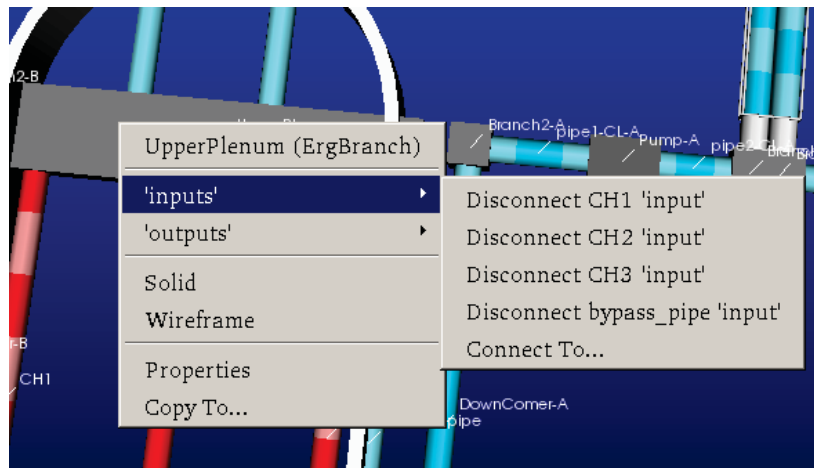


Figure 14: Component Context Menu Example

From the component context menu is also possible to change the visualization mode (wireframe or solid), bring up the corresponding parameters for editing, and make identical copies.

3.2 Control logic implementation

After the generation of the input file of the PWR model through the GUI, the following step is the implementation of the control logic that drives the sequence of events characterizing a SBO scenario:

- At 100 s, the transient begins (restart from steady state conditions previously computed)
- At 101 s, grid power is lost and immediate shutdown of the reactor occurs (scram), followed by:
 - Pump coast down
 - Decay heat power
 - Diesel Generators fail to operate (AC power lost) and, thus, auxiliary cooling system is inoperable
- At 1675 s, recovery of the AC power; auxiliary cooling system starts operating
- At 2500 s, transient ends (No Clad failure)

Figure 15 and Figure 17 show the procedure to create controlled and monitored variables. In each RELAP-7 component, the controllable and ‘monitored’ variables are listed in the *controlled* and *monitored* drop down menu in each component. Figure 15 shows how to create the controlled variable that is linked to the Head of the Pump-A. Selecting from the corresponding controlled drop down menu, the user can create a variable chosen among the listed ones. The procedure to create a monitored variable is similar to the previous one, but, since a monitored variable is the result of an operation on a RELAP-7 field variable, the user needs to also specify an operator that will be applied on it during the calculation. In Figure 17, for example, the “Nodal Maximum Value” operator, applied to the clad heat structure, produces, as result, the maximum temperature in the clad.

The auxiliary variables are created in the *RavenAuxiliary* input block. Figure 17 shows the creation of the Raven Auxiliary variable *scram_start_time* that is used as a signal at the beginning of the accident sequence.

In order to perform the SBO analysis, some ‘tools’ (so named in the code), aimed to emulate different component behaviors, are needed. As already mentioned, after the loss of power grid and consequent immediate scram of the reactor, the primary and secondary pumps start the coast down and the reactor power is driven by the decay heat.

All these dynamic evolutions are described by the *RavenTools* block, in which the user can choose among different pre-defined “functions”. Figure 18 shows how the Pump Coast-Down tool has been created. All tools are registered in the MOOSE software *factory* (indicates a software structure used to standardize function interfaces); therefore, any time a new one is added, it appears automatically as a possible entry in the GUI associated block.

As can be inferred by the control logic input reported in Appendix A, all the Raven variables, tools, etc. are available and usable in the python environment with a prefix that indicate their type (e.g. controlled, monitored) and a name that is the one assigned by the user at the input stage.

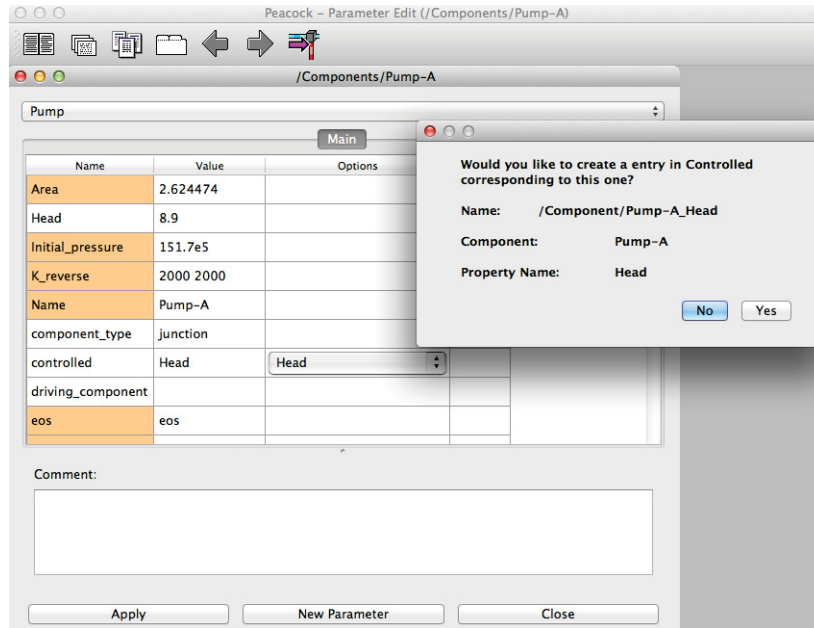


Figure 15: Generation of a controlled variable

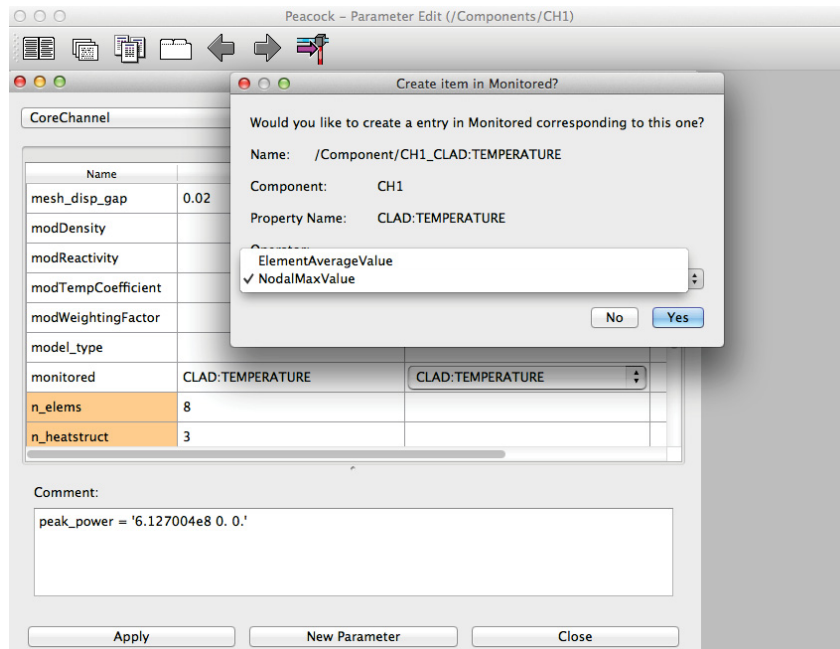


Figure 16: Generation of a monitored variable

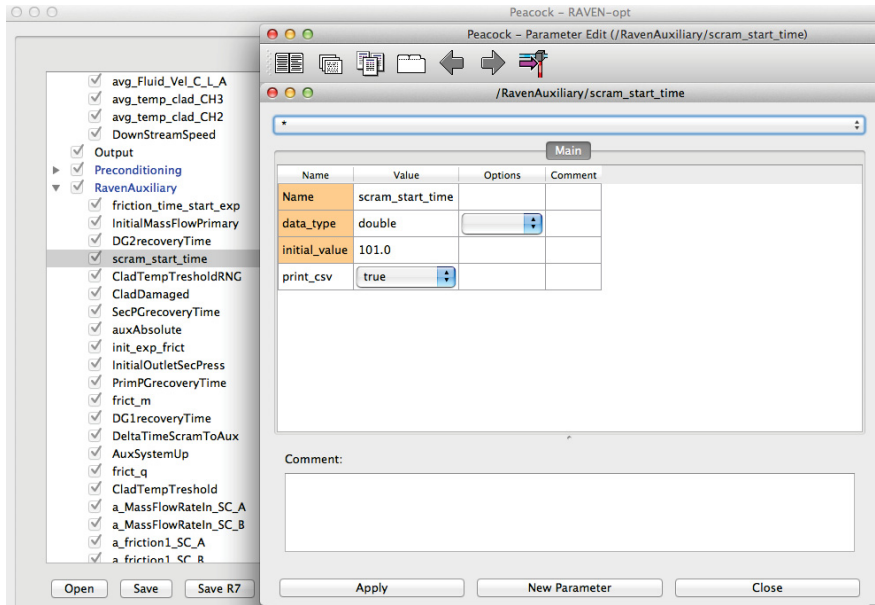
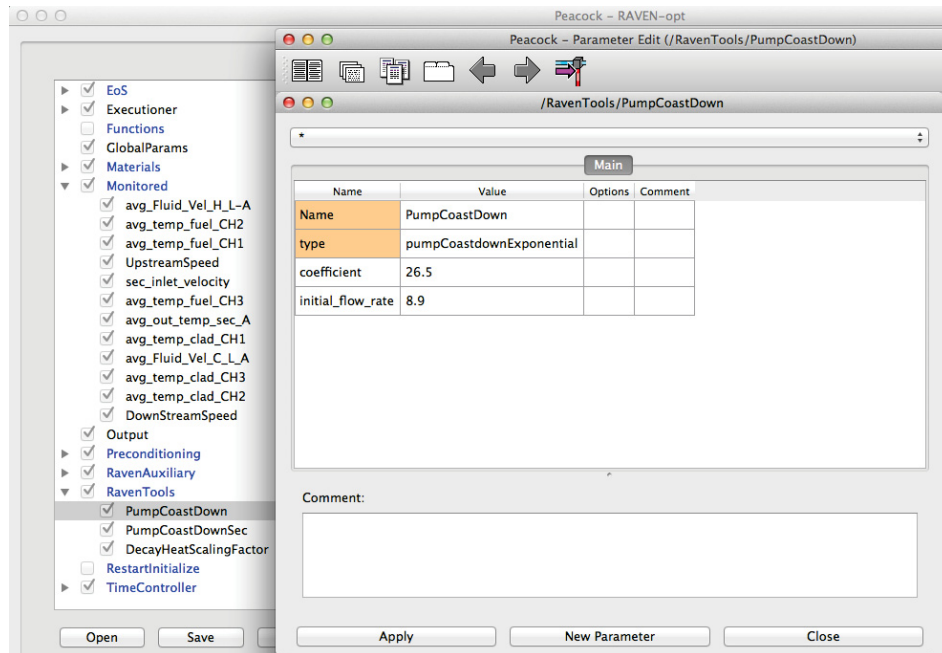


Figure 17: Generation of an auxiliary variable



3.3

Figure 18: Tools creation

3.4 Online Monitoring

The user can follow the evolution of the simulation in two tabs: Postprocessors and Visualize that are shortly illustrated below.

3.4.1 Postprocessors

In this GUI tab, which main functions are summarized in Figure 19, the user can follow the evolution of the auxiliary, monitored and controllable variables. Since this operation involves the reading of an optional file from the RELAP-7 code, it requires setting a proper logical flag (`print_csv`) for each variable that the user wants to visualize. The displayable variables are listed on the left side of the tab and when one of these is activated the corresponding plot is shown in the display window. Multiple plots are sequentially arranged.

3.4.2 Visualize

The tab named “Visualize” (see Figure 20) allows the visualization of the solution fields computed by RELAP-7. On the right side there is a list of the components that could be possibly activated and the different solution fields present in the actual simulation (e.g. solid temperature, fluid temperature, pressure, etc.).

On the bottom of the tab there are switches that can be used by the user for controlling the execution of the movie representing the time evolution of the selected fields and components. Like in the input tab, there are commands to handle and customize the plant layout 3D visualization (e.g. zoom, shift, rotation, etc.).

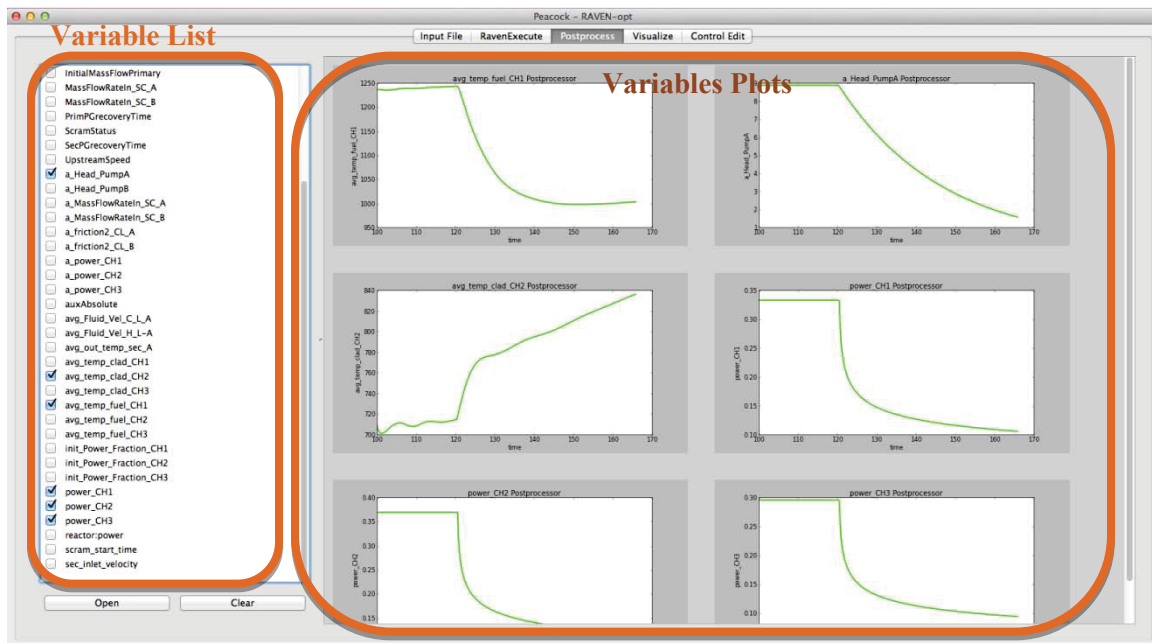


Figure 19: Postprocessor tab

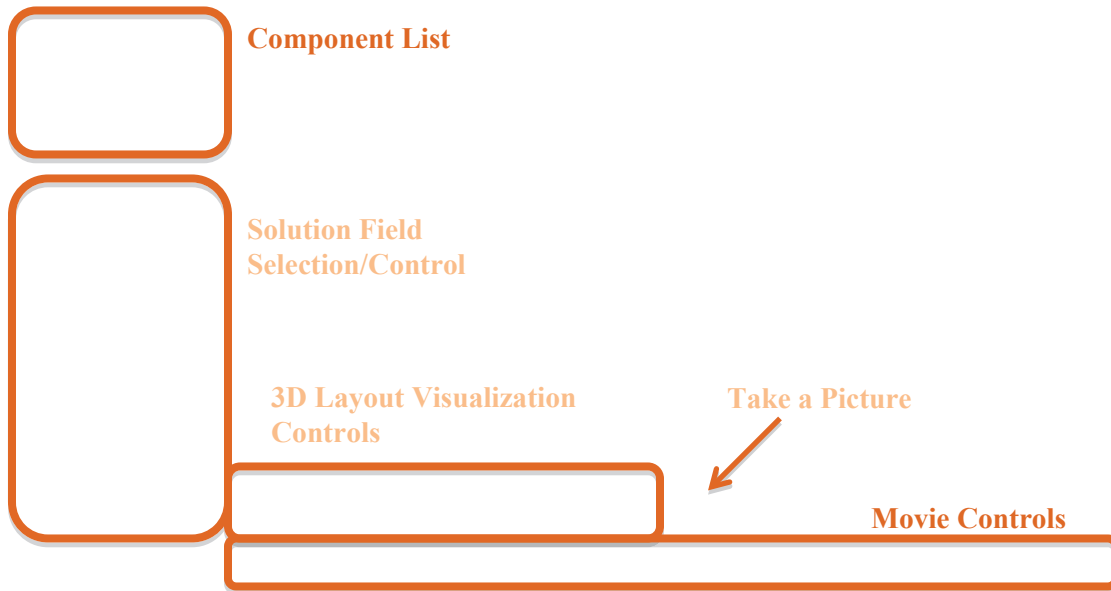


Figure 20: Visualize tab

3.5 Result Analysis

In this section, the results for the reference case will be discussed.

As already mentioned, the accident sequence begins when the Reactor is in normal operation state (reference point set at 100 seconds). At 101 seconds the external power grid is lost and the reactor scrams. All the auxiliary cooling systems and the Diesel generators fail to operate bringing the nuclear power plant in SBO state.

For this reference analysis, the average recovery times for trains of Diesel Generators, Primary and Secondary Power Grid have been used. As a results the Auxiliary Cooling System is back on line at time = 1675 seconds.

Figure 21 shows the Pump-A/B Head evolution. Right after the loss of power, the Pumps in the two primary loops begin an exponential coast down until the flow regime reaches the natural circulation mode. At 1675 seconds, the cooling system is recovered and the Head of the pumps reaches (rump-up ~ 1 s) the 5% of the operational value (8.9 m).

Figure 22 shows the temperature of the clad in the hot channel (Channel 1). Right after the reactor scram, the temperature of the clad starts rising until the auxiliary cooling system is restored. As can be noticed, the backup of the cooling system causes oscillations on the clad temperature. These oscillations are determined by the sudden insertion of the auxiliary system that imposes a mass flow rate ~ 5% of the operational one. The initial local maximum (~150 sec) is due to the sudden loss of heat sink while the fuel still behaves as a heat reserve; after a while the remaining cooling capacity (pumps' inertia) succeeds in diminishing the temperature (between 150 and 500 sec). The situation inverses again when the primary and secondary pumps completely stop.

It must be noted that the transient has been shortened in order to contain the computational time and because the main goal of this analysis is to show the abilities of RAVEN performing PRA analysis rather than a full realistic SBO analysis.

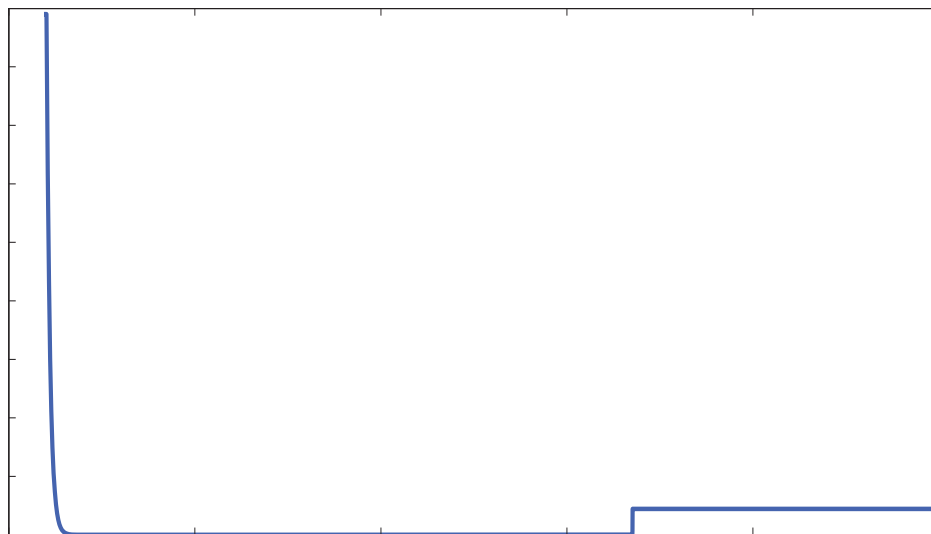


Figure 21: Pump A (equal pump B) head evolution

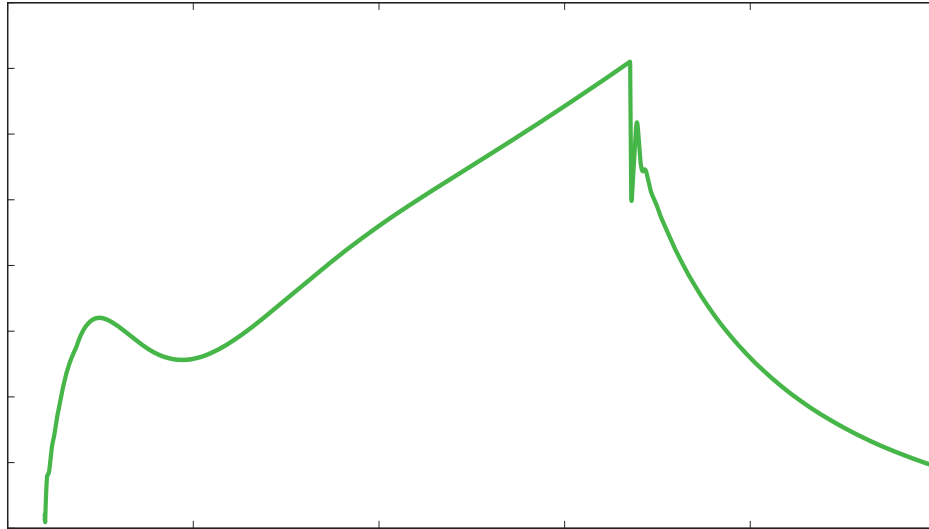


Figure 22: Average clad temperature in channel 1

4. Introducing Statistical Behaviors

This section presents the extension of the previously presented test case to show the capabilities of RAVEN to perform PRA.

4.1 PWR SBO Test Case

The system considered is a simplified model of a PWR system described in the previous section.

In order to simulate a SBO initiating event we added several components in the control logic (see Figure 23):

- Set of three diesel generators (DGs) and associated emergency buses
- Primary power grid line 138 KV (connected to the NSST switchyard)
- Auxiliary power grid line 69 KV (connected to the RSST switchyard)
- Electrical buses: 4160 V (step down voltage from the power grid and voltage of the electric converter connected to the DGs) and 480 V for actual reactor components (e.g., reactor cooling system)

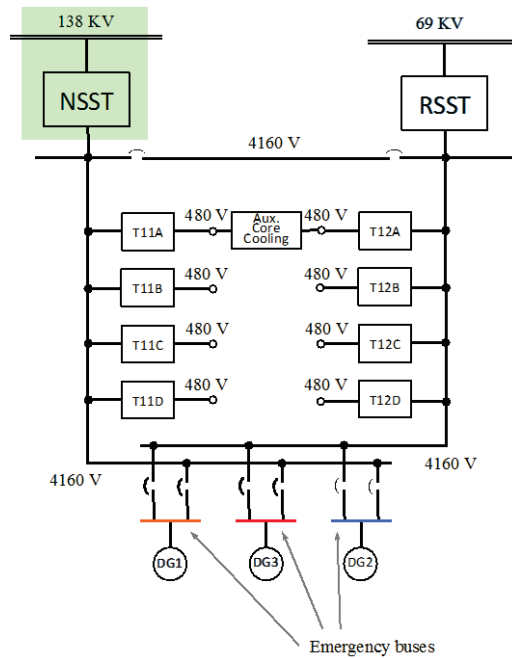


Figure 23: Scheme of the electrical system of the PWR model

The accident scenario is the following:

1. An external event causes a loss of off-site power (LOOP) due to damage of the 138 KV line and RSST switchyard; the reactor successfully scrams and, thus, power generated in the core follows the characteristic exponential decay curve
2. The set of DGs fail to start and, hence, conditions of SBO are reached (4160 V and 480 V buses are not energized); all cooling systems are subsequently off-line
3. Without the ability to cool the reactor core, its temperature starts to rise
4. In order to recover AC electric power on the 4160 V and 480 V buses, two recovery teams are assembled with the following strategy:
 - a. Recovery Team 1 focuses on the recovery of the DGs: due to internal damage at the DG building, two DGs (i.e., DG1 and DG3) need to be repaired (see Figure 24(a))
 - b. Recovery Team 2 focuses on the recovery of the RSST switchyard; 69KV line is energized but the RSST switchyard needs to be recovered (see Figure 24(a))
5. Meanwhile the owning company is working on the restoration of the primary 138 KV line (see Figure 24(a))
6. When the 4160 KV buses are energized (through the recovery of the DGs, RSST or 138KV line), the auxiliary cooling system is able to cool the reactor core and, thus, core temperature decreases.

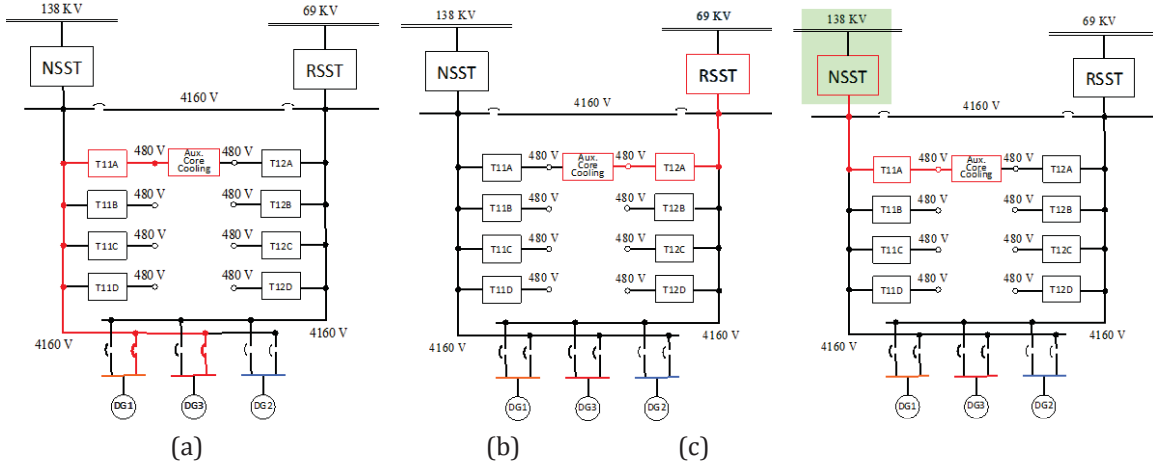


Figure 24: AC power recovery paths through: DGs (a), RSST (b) and 138KV line (c). Red lines indicate electrical path to power Auxiliary cooling system

4.2 Modeling

Given the uncertainties associated to the recovery of both DGs, RSST and 138KV line, we modeled these three recovery events as stochastic events with probabilistic distributions associated to them. Given the time scale associated to the dynamics of the RELAP7 PWR model the distribution were chosen as follows (see Figure 25):

- DGs: a dead time of 100s is required by Team 1 to gather at the DGs building and DG1 repair time T_{DG1} has a normal distribution having $\mu = 800$ and $\sigma = 200$. This distribution is also truncated such that $0 < T_{DG1} < 2500$. The recovery time of DG3, T_{DG3} , is proportional to T_{DG1} . Such relation has modeled using a multiplication factor T_{12} , i.e., $T_{DG3} = T_{DG1} \cdot T_{12}$. T_{12} is uniformly distributed between $[0.5 \ 1]$
- RSST: a dead time of 400s is needed to assess the damage at the RSST switchyard and to plan its recovery. Recovery time for RSST, T_{RSST} , is normally distributed with $\mu = 1400$ and $\sigma = 400$
- 138KV line: the recovery of the main AC line T_{138} is normally distributed with $\mu = 2000$ and $\sigma = 500$

In addition, the clad failure temperature $T_{C,fail}$ is not fixed but it is probabilistic distributed with a triangular distribution characterized by the following parameters:

- mode: $x_{Peak} = 2200 \text{ F} (1477.59 \text{ K})$, 10CFR regulatory limit
- lower bound: $x_{Min} = 1800 \text{ F} (1255.37 \text{ K})$, PRA success criterion
- upper bound: $x_{Max} = 2600 (1699.82 \text{ K})$, Urbanic-Heidrick transition temperature

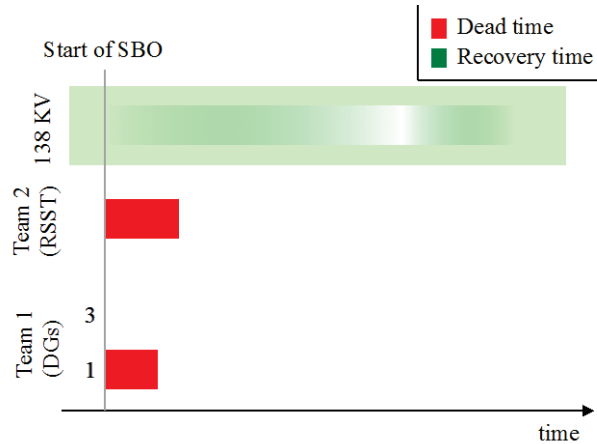


Figure 25: Recovery timings for DGs, RSST and 138 KV line (color intensity is proportional to probability)

Thus, the Monte-Carlo analysis, for each run consist of the following steps:

- Sample the values of: T_{DG1} , T_{DG3} , T_{RSST} , T_{138} and $T_{C,fail}$ from their own distribution
- Perform a simulation run; under SBO conditions the clad temperature rises
 - If AC power is recovered (through DGs, RSST or 138 KV) then cooling is restored and clad temperature starts to decrease
 - If temperature of the clad reaches $T_{C,fail}$, then core damage occurs

4.3 Results

We performed a total of 4000 simulation runs. With such high number of runs, it was possible to obtain a good statistic on the five stochastic variables and, thus, also on the output variables such as the maximum clad temperature reached inside the core. In this respect, Figure 26 and Figure 27 show the obtained distributions for the five stochastic variables: $T_{C,fail}$, T_{DG1} , T_{DG3} , T_{RSST} , T_{138} . Figure 28 shows the temporal profiles of the clad temperature inside CH1 for all simulation runs.

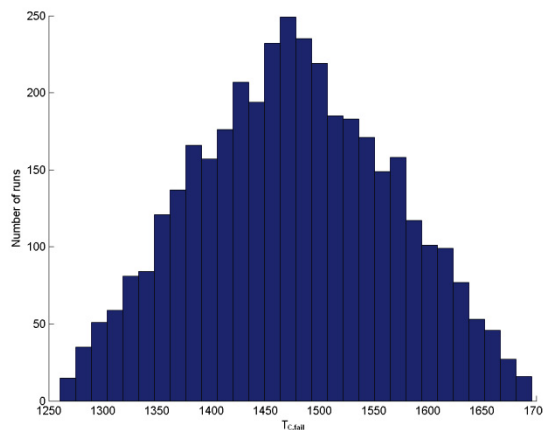


Figure 26: Distribution profiles obtained from the 4000 sample for $T_{C,fail}$

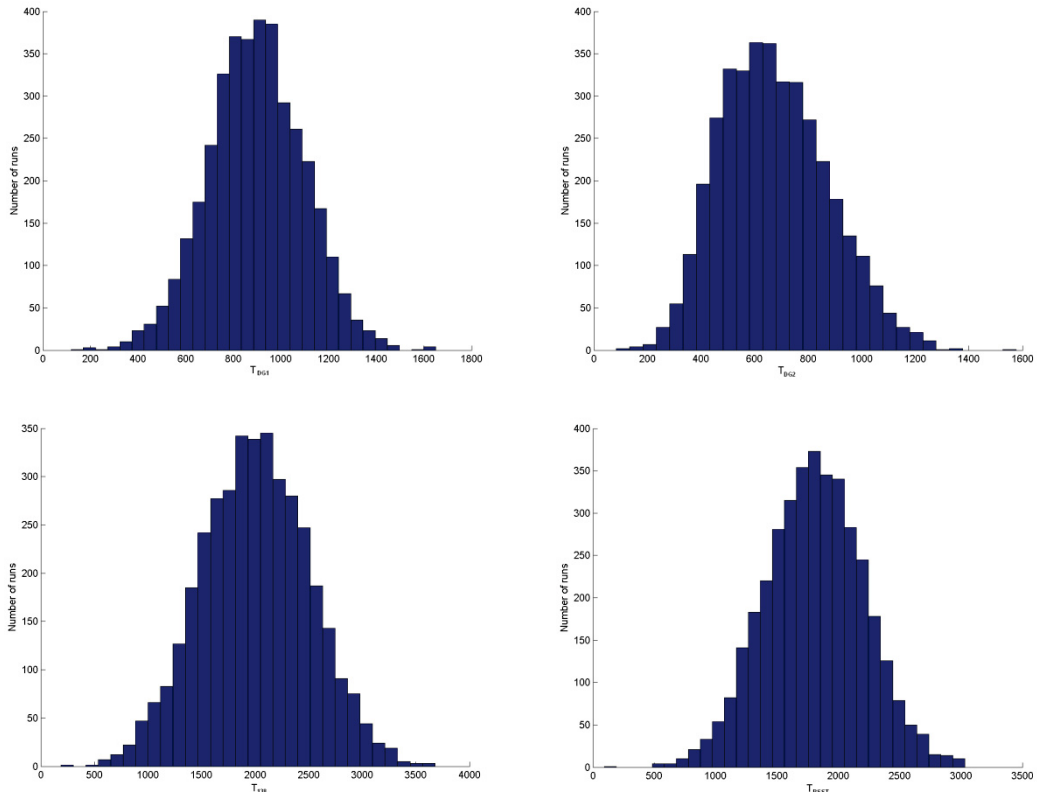


Figure 27: Distribution profiles obtained from the 4000 sample for T_{DG1} (top left), T_{DG3} (top right), T_{RSST} (bottom left) and T_{138} (bottom right)

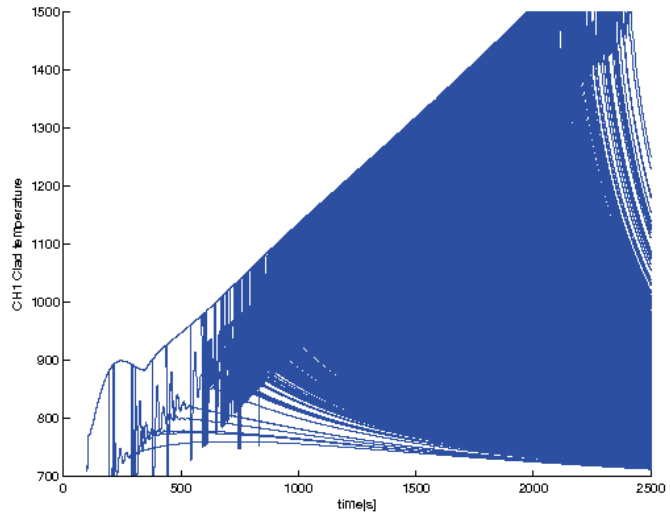


Figure 28: Temporal profiles of the clad temperature in CH1 for all the 4000 simulations

In order to analyze the data set obtained, we compared the distribution of the clad failure temperature with the maximum temperature reached inside the core (see Figure 29).

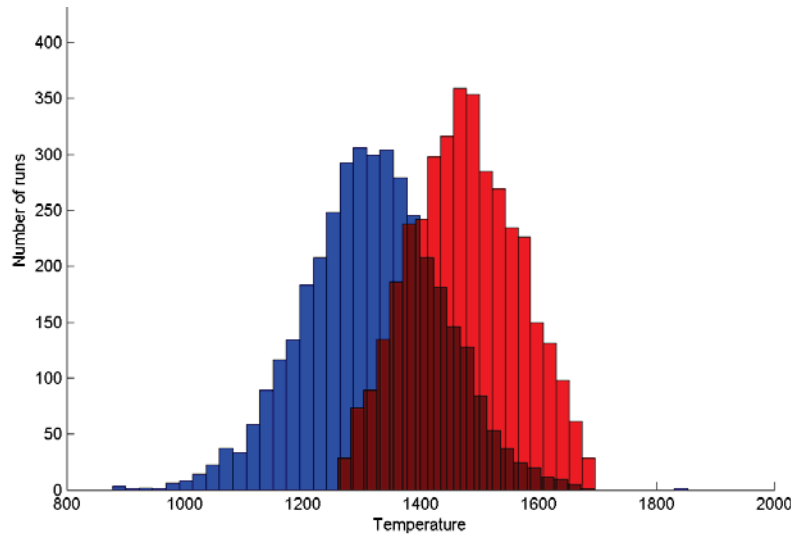


Figure 29: Distribution of clad failure temperature (red) and maximum clad temperature reached in the simulation (blue)

The region when the two distributions overlap indicates the range of temperature where failure might occur.

When this effect is accounted for, that is the simulation is stopped when the temperature of the clad failure is reached (accounting for his random value), the histogram of the maximum temperature changes as shown Figure 30. The distortion seen with respect the Figure 29 is in agreement with the mathematical explanation given in the next section of the analysis of results.

We then performed a series of exercises to identify how the 5 stochastic variables contribute to system failure.

A first approach was to identify how uncertainties associated to clad failure temperature and recovery time of auxiliary cooling system would affect simulation outcome (i.e., failure or success). In particular, we determined the limit surface, i.e., the boundaries in this 2-D space between system failure and success.

Figure 8 shows such boundaries along which the scatter plot of all 4000 simulation runs pictured in green or red depending on simulation outcome (success or a failure respectively). The limit surface has been determined using a Support Vector Machine classifier as shown in [14].

As expected, system failure occurs for low values of clad fail temperature and high values of auxiliary cooling system recovery time (i.e., when AC power is recovered through DGs, RSST or 138KV line).

We also extended the evaluation of the limit surface on a 3-D case by considering only the recovery time of DGs, RSST and 138 KV line and keeping a fixed value for $T_{C,fail} = 2200$ F or 1477 K (i.e., we are looking at a projection of the actual 4-D limit surface in a 3-D space). Such limit surface is shown in Figure 32. For such specific value of $T_{C,fail}$, the minim value for recovery time of the auxiliary cooling system is about 1800s (also confirmed in Figure 31). Thus in order to obtain system success, a minimum value of 1800s is needed for at least one of the three recoveries events. This explains the geometrical shape of the limit surface shown in Fig. Figure 32.

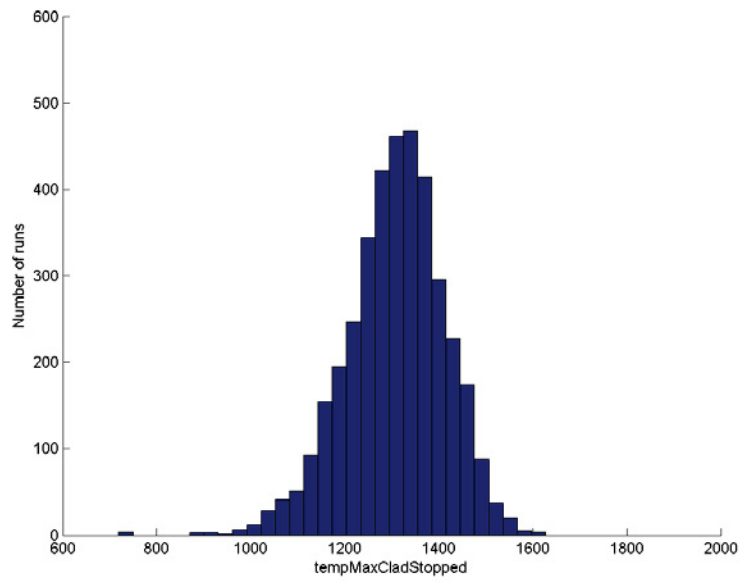


Figure 30: Max clad temperature histogram when failure is accounted for

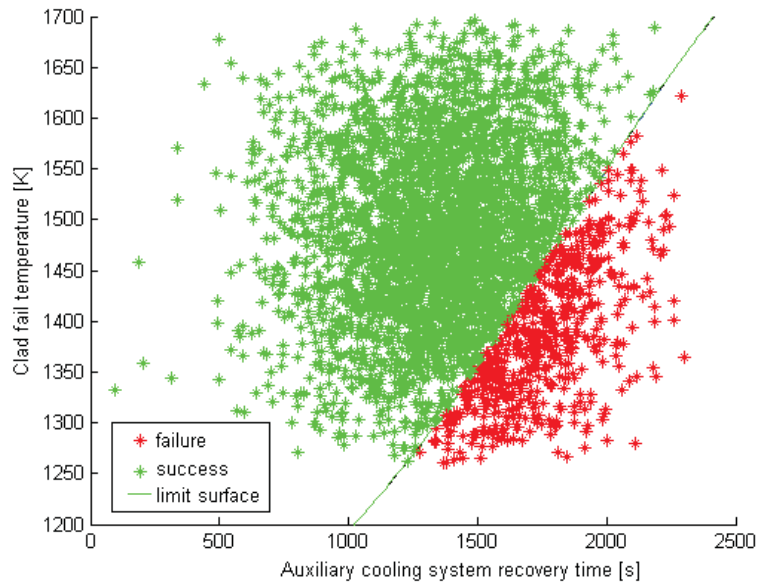


Figure 31: 2D limit surface

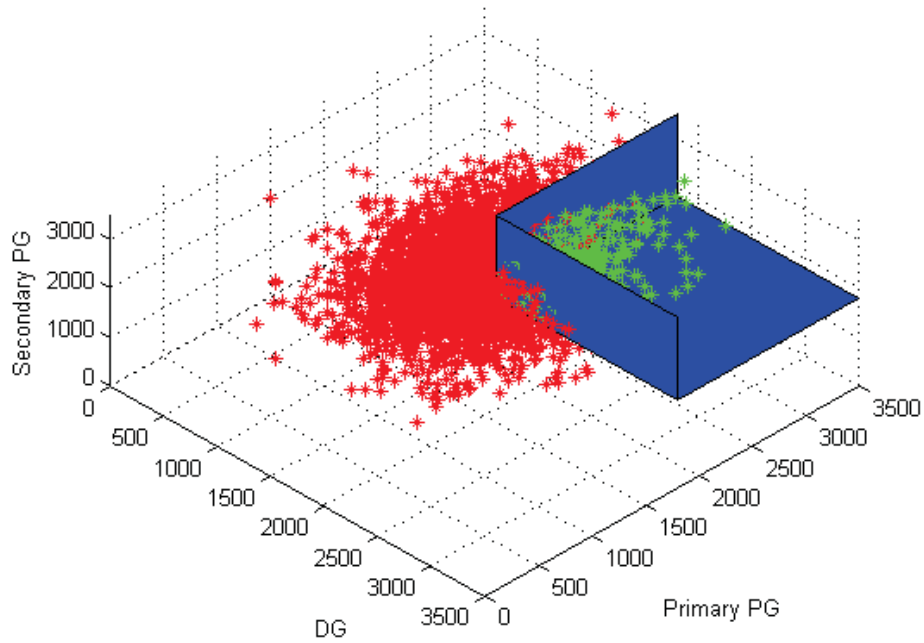


Figure 32: 3D limit surface

4.3.1 Explanation of the Effect on the Maximum Clad Temperature Using a Random Distribution on the Failure Temperature

As already explained, the Figure 30 shows the maximum clad temperature achieved during the simulation if the simulation gets stopped in case of clad failure (i.e. the clad temperature exceeds the failure value).

To better understand the results, it is possible to analyze a simplified case:

- The maximum clad temperature without failing any simulation before reaching the end mission time (the user given time value for which the simulation stops) has a known Gaussian shape (this is an assumption not an outcome from the simulation): $T_{Max} \cong G_{T_{Max}}(\mu = 1400, \sigma = 100)$
- The clad failure temperature has a Gaussian shape as well (in the simulation a triangular shape has been used instead): $T_F \cong G_{T_F}(\mu = 1500, \sigma = 50)$

The probability distribution of the maximum temperature achieved by the clad, when considering both effects, has then two contribution:

$$T_{Max,C} \cong G_{T_{Max}}(T_{Max,C}) \left(1 - \int_{-\infty}^{T_{Max,C}} dT_F G_{T_F}(T_F) \right) + G_{T_F}(T_{Max,C}) \left(1 - \int_{T_{Max}}^{\infty} dT_{Max} G_{T_{Max}}(T_{Max}) \right)$$

Eq. 8

Where the first term accounts for the probability to have as a maximum temperature $T_{Max,C}$ decreased by the probability that the failure temperature was below such a value.

The second term instead accounts for the probability that the maximum temperature would have been above $T_{Max,C}$ while the failure temperature exactly equal to $T_{Max,C}$. This effect is graphically illustrated in Figure 33. This analysis explains the distortion in the distribution seen comparing Figure 29 and Figure 30 in the simulation results analysis

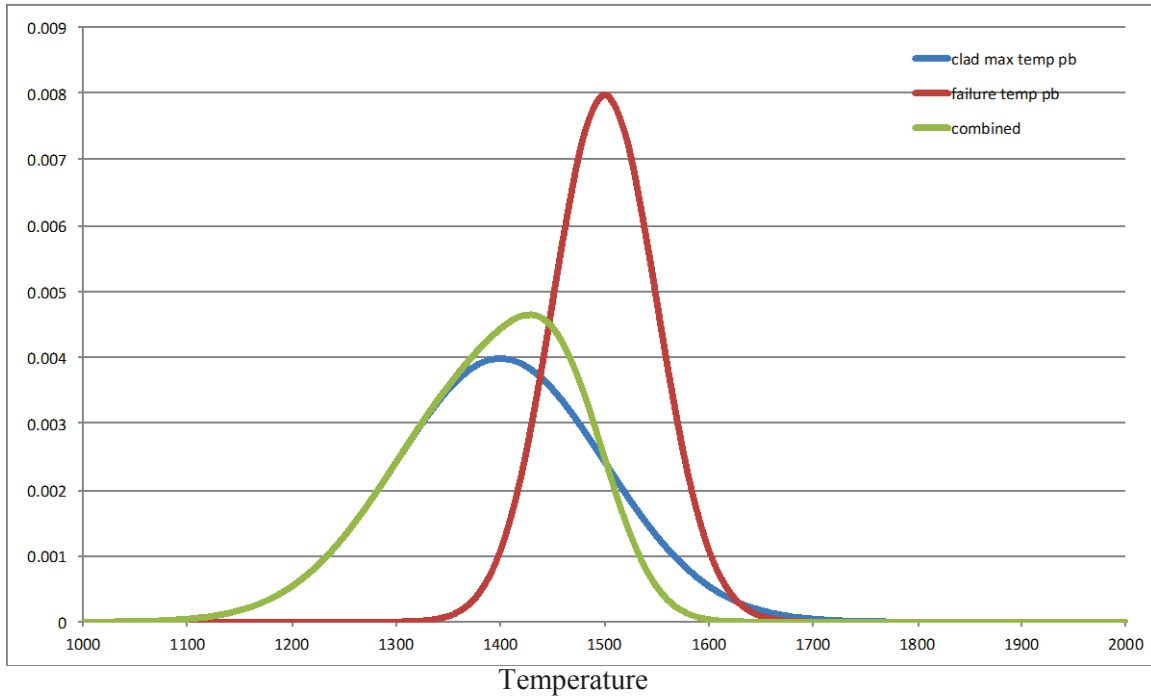


Figure 33: Combined effect of the uncertainty in the temperature failure on the maximum clad temperature recorded by the simulation

5. Conclusion

The milestone of performing a PRA demo on a PWR reactor has been fully achieved and has been described in this report. In addition to the technical achievements, a framework that makes this results replicable and stable has been generated. The framework is capable to run hundreds of simulations in parallel and handle the data post processing. Data are currently stored in HDF5 compressed format, which is a general portable format supported in almost all computing architectures [15]. The limit surface approach, while still under development (adaptive loop not yet in place), has shown strong potential to reduce the computational load of the Monte Carlo approach.

RAVEN is moving forward to provide all the tools needed by the RISMC conceptual framework to bring the promised innovations to the risk management practice in the nuclear industry.

There is additional work that still needs to be done:

- The artificial intelligence aided discovery framework still needs to be integrated within the RAVEN/Peacock GUI
- The iterative process to take full advantage of ROM predictive capabilities has yet to be permanently added to RAVEN
- A performance analysis should be done on the several ROM models provided by scikit-learn on more complex cases and possibly derive new ones specific to the type of problems of interest to RISMC

- Tools to compute the sensitivity coefficients on the risk function should be derived and implemented in order to generate the needed information to initiate optimization (i.e. reducing risk within economical constrain)
- The implementation of data mining (unsupervised learning) to identify and rank risk sources has not yet started

Nonetheless RAVEN, after a year and half of development, is showing a clear path forward, the fundament of the software infrastructure has been built and has begun to show its value to the LWRS program.

In conclusion, it is likely that the project will deliver its promises and will become key to the RISMC toolkit.

5.1 References

1. "Light Water Reactor Sustainability Program Integrated Program Plan, Revision 1," INL-EXT-11-23452, April 2013
2. R. W. Youngblood, V. A. Mousseau, D. L. Kelly, and T.N. Dinh, "Risk-Informed Safety Margin Characterization (RISMC): Integrated Treatment of Aleatory and Epistemic Uncertainty in Safety Analysis," *The 8th International Topical Meeting on Nuclear Thermal-Hydraulics, Operation and Safety (NUTHOS-8)* Shanghai, China, October 10-14, 2010
3. "RELAP-7 Level 2 Milestone Report: Demonstration of a Steady State Single Phase PWR Simulation with RELAP-7," INL/EXT-12-25924
4. D. Gaston, C. Newman, G. Hansen, D. Lebrun- Grandié, "MOOSE: A parallel computational framework for coupled systems of nonlinear equations", *Nuclear Engineering and Design*, Vol. 239, Issue 10, pp. 1768- 1778 (2009).
5. R. L. Williamson, J.D. Hales, S.R. Novascone, M.R. Tonks, D.R. Gaston, C.J. Permann, D. Andrs, R.C. Martineau, "Multidimensional Multiphysics Simulation of Nuclear Fuel Behavior," *Journal of Nuclear Materials*, Vol. 423, pp. 149-163, 2012
6. B. Spencer, J. Busby, R. Martineau, B. Wirth "A Proof of Concept: Grizzly, the LWRS Program Materials Aging and Degradation Pathway Main Simulation Tool," INL/EXT-12-27559
7. F. N. Gleicher II, Y. Wang, D. Gaston, R. C. Martineau, "The Method of Manufactured Solutions for RattleSnake, A SN Radiation Transport Solver inside the MOOSE Framework," *proc. Am. Nuc. Soc.*, Chicago, IL 2012
8. C. Rabiti, A. Alfonsi, J. Cogliati, D. Mandelli, R. Kinoshita "Reactor Analysis and Virtual Control Environment (RAVEN) FY12 Report," INL/EXT-12-27351
9. C. Rabiti, A. Alfonsi, D. Mandelli, J. Cogliati, R. Kinoshita "MATHEMATICAL FRAMEWORK FOR THE ANALYSIS OF DYNAMC STOCHASTIC SYSTEMS WITH THE RAVEN CODE," International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering May 4, 2013
10. B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey. libMesh: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations. *Engineering with Computers*, 22(3-4):237-254, 2006.
11. S. Balay, J. Brown, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C.

- McInnes, B. F. Smith, H. Zhang "PETSc Web page," <http://www.mcs.anl.gov/petsc> 2013.
12. F. Pedregosa, A. Gramfort, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research 12 (2011) 2825-2830
 13. "Pressurized Water Reactor Main Steam Line Break (MSLB) Benchmark", Volume I: Final Specifications, NEA/NSC/DOC(99)8.
 14. D. Mandelli and C. Smith, "Adaptive sampling using support vector machines," in Proceeding of American Nuclear Society (ANS), San Diego (CA), vol. 107, pp. 736-738, 2012
 15. The HDF Group. Hierarchical data format version 5, 2000-2010. <http://www.hdfgroup.org/HDF5>.

6. APPENDIXES

6.1 APPENDIX A: Station Black Out Inputs

6.1.1 Plant and Raven Tools Input

```
[GlobalParams]
model_type = 3
global_init_P = 15.17e6
global_init_V = 0.
global_init_T = 564.15
scaling_factor_var = '1.e-1 1.e-5 1.e-8'
[]
[EoS]
[./eos]
e_0 = 3290122.80 # J/kg
beta = .46e-3 # K^{-1}
a2 = 1.e7 # m^2/s^2
rho_0 = 738.350 # kg/m^3
T_0 = 564.15 # K
type = NonIsothermalEquationOfState
cv = 5.832e3 # J/kg-K
p_0 = 15.17e6 # Pa
[./]
[]
[Materials]
[./fuel-mat]
k = 3.65
Cp = 288.734
type = SolidMaterialProperties
rho = 1.032e4
[./]
[./gap-mat]
k = 1.084498
Cp = 1.0
type = SolidMaterialProperties
rho = 1.
[./]
[./clad-mat]
k = 16.48672
Cp = 321.384
type = SolidMaterialProperties
rho = 6.55e3
[./]
[./wall-mat]
k = 1.0
Cp = 4.0
type = SolidMaterialProperties
rho = 80.0
[./]
[]
[Components]
[./reactor]
initial_power = 2.77199979e9
type = Reactor
[./]
[./CH1]
elem_number_of_hs = '3 1 1'
Ts_init = 564.15
orientation = '0 0 1'
rho_hs = '1.0412e2 1.0 6.6e1'
aw = 276.5737513
n_elems = 8
k_hs = '3.65 1.084498 16.48672'
material_hs = 'fuel-mat gap-mat clad-mat'
Dh = 0.01332254
fuel_type = cylinder
name_of_hs = 'FUEL GAP CLAD'
Hw = 5.33e4
n_heatstruct = 3
A = 1.858983051
power_fraction = '2.96405926e-1 0 0'
f = 0.01
type = CoreChannel
Cp_hs = '288.734 1.0 6.6e3'
eos = eos
length = 3.6576
position = '0 1.2 0'
width_of_hs = '0.0046955 0.0000955 0.000673'
[./]
[./bypass_pipe]
A = 1.589571014
Hw = 5.33e4
n_heatstruct = 3
A = 1.161864
power_fraction = '3.33672612e-1 0 0'
f = 0.01
type = CoreChannel
Cp_hs = '288.734 1.0 321.384'
eos = eos
length = 3.6576
position = '0 -1.2 0'
width_of_hs = '0.0046955 0.0000955 0.000673'
[./]
[./CH2]
elem_number_of_hs = '3 1 1'
Ts_init = 564.15
orientation = '0 0 1'
rho_hs = '1.0412e2 1.0 6.6e1'
aw = 276.5737513
n_elems = 8
k_hs = '3.65 1.084498 16.48672'
material_hs = 'fuel-mat gap-mat clad-mat'
Dh = 0.01332254
fuel_type = cylinder
name_of_hs = 'FUEL GAP CLAD'
Hw = 5.33e4
n_heatstruct = 3
A = 1.549152542
power_fraction = '3.69921461e-1 0 0'
f = 0.01
type = CoreChannel
Cp_hs = '288.734 1.0 321.384'
eos = eos
length = 3.6576
position = '0 0 0'
width_of_hs = '0.0046955 0.0000955 0.000673'
[./]
[./CH3]
elem_number_of_hs = '3 1 1'
Ts_init = 564.15
orientation = '0 0 1'
rho_hs = '1.0412e2 1.0 6.6e1'
aw = 276.5737513
n_elems = 8
k_hs = '3.65 1.084498 16.48672'
material_hs = 'fuel-mat gap-mat clad-mat'
Dh = 0.01332254
fuel_type = cylinder
name_of_hs = 'FUEL GAP CLAD'
Hw = 5.33e4
n_heatstruct = 3
A = 1.858983051
power_fraction = '2.96405926e-1 0 0'
f = 0.01
type = CoreChannel
Cp_hs = '288.734 1.0 6.6e3'
eos = eos
length = 3.6576
position = '0 1.2 0'
width_of_hs = '0.0046955 0.0000955 0.000673'
[./]
```

```

orientation = '0 0 1'
Dh = 1.42264
f = 0.001
Hw = 0.0
eos = eos
length = 3.6576
n_elems = 5
position = '0 1.5 0'
type = Pipe
[./]
[./LowerPlenum]
inputs = 'DownComer-A(out) DownComer-B(out)'
Area = 3.618573408
outputs = 'CH1(in) CH2(in) CH3(in) bypass_pipe(in)'
K = '0.2 0.2 0.2 0.2 0.4 40.0'
eos = eos
Initial_pressure = 151.7e5
type = ErgBranch
[./]
[./UpperPlenum]
inputs = 'CH1(out) CH2(out) CH3(out) bypass_pipe(out)'
Area = 7.562307456
outputs = 'pipe1-HL-A(in) pipe1-HL-B(in)'
K = '0.5 0.5 0.5 80.0 0.5 0.5'
eos = eos
Initial_pressure = 151.7e5
type = ErgBranch
[./]
[./DownComer-A]
A = 3.6185734
orientation = '0 0 -1'
Dh = 1.74724302
f = 0.001
Hw = 0.
eos = eos
length = 4
n_elems = 3
position = '0 2.0 4.0'
type = Pipe
[./]
[./pipe1-HL-A]
A = 7.562307456
orientation = '0 0 1'
Dh = 3.103003207
f = 0.001
Hw = 0.0
eos = eos
length = 4.
n_elems = 3
position = '0 0.5 4.0'
type = Pipe
[./]
[./pipe2-HL-A]
A = 2.624474
orientation = '0 1 0'
Dh = 1.828
f = 0.001
Hw = 0.0
eos = eos
length = 3.5
n_elems = 3
position = '0 0.5 8.0'
type = Pipe
[./]
[./pipe1-CL-A]
A = 2.624474
orientation = '0 -1 0'
Dh = 1.828
f = 0.001
Hw = 0.0
eos = eos
length = 1.
n_elems = 3
position = '0 3.0 4.0'
type = Pipe
[./]
[./pipe2-CL-A]
A = 2.624474
orientation = '0 -1 0'
Dh = 1.828
f = 0.001
Hw = 0.0
eos = eos
length = 0.8
n_elems = 3
position = '0 4 4.0'
type = Pipe
[./]
[./pipe1-SC-A]
A = 1.3122
orientation = '0 -1 0'
Dh = 0.914
f = 0.001
Hw = 0.0
eos = eos
length = 1.
n_elems = 3
position = '0 5.2 4.0'
type = Pipe
[./]
[./pipe2-SC-A]
A = 1.3122
orientation = '0 1 0'
Dh = 0.914
f = 0.001
Hw = 0.0
eos = eos
length = 1.
n_elems = 3
position = '0 4.2 8.0'
type = Pipe
[./]
[./Branch1-A]
inputs = pipe1-HL-A(out)
Area = 7.562307456
outputs = 'pipe2-HL-A(in) pipe-to-Pressurizer(in)'
K = '0.5 0.7 80.'
eos = eos
Initial_pressure = 151.7e5
type = ErgBranch
[./]
[./Branch2-A]
inputs = pipe1-CL-A(out)
Area = 3.6185734
outputs = DownComer-A(in)
K = '0.5 0.7'
eos = eos
Initial_pressure = 151.7e5
type = ErgBranch
[./]
[./Branch3-A]
inputs = pipe2-HL-A(out)
Area = 2.624474
outputs = HX-A(primary_in)
K = '0.5 0.7'
eos = eos
Initial_pressure = 151.7e5
type = ErgBranch
[./]
[./Pump-A]

```

```

inputs = pipe2-CL-A(out)
Head = 8.9
Area = 2.624474
outputs = pipe1-CL-A(in)
eos = eos
Initial_pressure = 151.7e5
K_reverse = '2000 2000'
type = Pump
[./]
[./HX-A]
orientation = '0 0 -1'
aw = 539.02
n_elems = 10
A_secondary = 5
material_wall = wall-mat
wall_thickness = 0.001
Dh = 0.01
Twall_init = 564.15
Hw = 1.e4
aw_secondary = 539.02
eos_secondary = eos
type = HeatExchanger
A = 5.0
Dh_secondary = 0.001
Hw_secondary = 1.e4
n_wall_elems = 2
Cp_wall = 100.0
f_secondary = 0.01
rho_wall = 100.0
k_wall = 100.0
f = 0.01
eos = eos
length = 4.
position = '0 4. 8.'
dim_wall = 1
[./]
[./Branch4-A]
inputs = pipe1-SC-A(out)
Area = 2.624474e2
outputs = HX-A(secondary_in)
K = '0.5 0.7'
eos = eos
Initial_pressure = 151.7e5
type = ErgBranch
[./]
[./Branch5-A]
inputs = HX-A(secondary_out)
Area = 2.624474e2
outputs = pipe2-SC-A(in)
K = '0.5 0.7'
eos = eos
Initial_pressure = 151.7e5
type = ErgBranch
[./]
[./Branch6-A]
inputs = HX-A(primary_out)
Area = 2.624474e2
outputs = pipe2-CL-A(in)
K = '0.5 0.7'
eos = eos
Initial_pressure = 151.7e5
type = ErgBranch
[./]
[./PressureOutlet-SC-A]
eos = eos
input = pipe2-SC-A(out)
p_bc = 151.7e5
type = TimeDependentVolume
T_bc = 564.15
[./]
[./DownComer-B]
A = 3.6185734
orientation = '0 0 -1'
Dh = 1.74724302
f = 0.001
Hw = 0.
eos = eos
length = 4
n_elems = 3
position = '0 -2.0 4.0'
type = Pipe
[./]
[./pipe1-HL-B]
A = 7.562307456
orientation = '0 0 1'
Dh = 3.103003207
f = 0.001
Hw = 0.0
eos = eos
length = 4.
n_elems = 3
position = '0 -0.5 4.0'
type = Pipe
[./]
[./pipe2-HL-B]
A = 2.624474
orientation = '0 -1 0'
Dh = 1.828
f = 0.001
Hw = 0.0
eos = eos
length = 3.5
n_elems = 3
position = '0 -0.5 8.0'
type = Pipe
[./]
[./pipe1-CL-B]
A = 2.624474
orientation = '0 1 0'
Dh = 1.828
f = 0.001
Hw = 0.0
eos = eos
length = 1.
n_elems = 3
position = '0 -3.0 4.0'
type = Pipe
[./]
[./pipe2-CL-B]
A = 2.624474
orientation = '0 1 0'
Dh = 1.828
f = 0.001
Hw = 0.0
eos = eos
length = 0.8
n_elems = 3
position = '0 -4.0 4.0'
type = Pipe
[./]
[./pipe1-SC-B]
A = 1.3122
orientation = '0 1 0'
Dh = 0.914
f = 0.001
Hw = 0.0
eos = eos
length = 1.
n_elems = 3
position = '0 -5.2 4.0'

```



```

type = Pipe
[./]
[/pipe2-SC-B]
A = 1.3122
orientation = '0 -1 0'
Dh = 0.914
f = 0.001
Hw = 0.0
eos = eos
length = 1.
n_elems = 3
position = '0 -4.2 8.0'
type = Pipe
[./]
[/Branch1-B]
inputs = pipe1-HL-B(out)
Area = 7.562307456
outputs = pipe2-HL-B(in)
K = '0.5 0.7'
eos = eos
Initial_pressure = 151.7e5
type = ErgBranch
[./]
[/Branch2-B]
inputs = pipe1-CL-B(out)
Area = 3.6185734
outputs = DownComer-B(in)
K = '0.5 0.7'
eos = eos
Initial_pressure = 151.7e5
type = ErgBranch
[./]
[/Branch3-B]
inputs = pipe2-HL-B(out)
Area = 2.624474
outputs = HX-B(primary_in)
K = '0.5 0.7'
eos = eos
Initial_pressure = 151.7e5
type = ErgBranch
[./]
[/Pump-B]
inputs = pipe2-CL-B(out)
Head = 8.9
Area = 2.624474
outputs = pipe1-CL-B(in)
eos = eos
Initial_pressure = 151.7e5
K_reverse = '2000 2000'
type = Pump
[./]
[/HX-B]
orientation = '0 0 -1'
aw = 539.02
n_elems = 10
A_secondary = 5 # 5.
material_wall = wall-mat
wall_thickness = 0.001
Dh = 0.01
Twall_init = 564.15
Hw = 1.e4
aw_secondary = 539.02 # 539.02
eos_secondary = eos
type = HeatExchanger
A = 5.
Dh_secondary = 0.001
Hw_secondary = 1.e4
n_wall_elems = 2
Cp_wall = 100.0
f_secondary = 0.01

rho_wall = 100.0
k_wall = 100.0
f = 0.01
eos = eos
length = 4.
position = '0 -4. 8.'
disp_mode = -1.0
[./]
[/Branch4-B]
inputs = pipe1-SC-B(out)
Area = 2.624474e2
outputs = HX-B(secondary_in)
K = '0.5 0.7'
eos = eos
Initial_pressure = 151.7e5
type = ErgBranch
[./]
[/Branch5-B]
inputs = HX-B(secondary_out)
Area = 2.624474e2
outputs = pipe2-SC-B(in)
K = '0.5 0.7'
eos = eos
Initial_pressure = 151.7e5
type = ErgBranch
[./]
[/Branch6-B]
inputs = HX-B(primary_out)
Area = 2.624474e2
outputs = pipe2-CL-B(in)
K = '0.5 0.7'
eos = eos
Initial_pressure = 151.7e5
type = ErgBranch
[./]
[/PressureOutlet-SC-B]
eos = eos
input = pipe2-SC-B(out)
p_bc = 151.7e5
type = TimeDependentVolume
T_bc = 564.15
[./]
[/pipe-to-Pressurizer]
A = 2.624474
orientation = '0 0 1'
Dh = 1.828
f = 10.
Hw = 0.0
eos = eos
length = 0.5
n_elems = 3
position = '0 0.5 8.0'
type = Pipe
[./]
[/Pressurizer]
eos = eos
input = pipe-to-Pressurizer(out)
p_bc = 151.7e5
type = TimeDependentVolume
T_bc = 564.15
[./]
[/MassFlowRateIn-SC-B]
v_bc = 2.542 # 4.542
input = pipe1-SC-B(in)
type = TimeDependentJunction
eos = eos
T_bc = 537.15
[./]
v_bc = 2.542 # 4.542
input = pipe1-SC-A(in)

```

```

type = TimeDependentJunction
eos = eos
T_bc = 537.15
[./]
[]
[Preconditioning]
active = 'SMP_PJFNK'
[./SMP_PJFNK]
petsc_options_iname = '-mat_fd_type -mat_mffd_type'
full = true
type = SMP
petsc_options_value = 'ds ds'
petsc_options = '-snes_mf_operator'
[./]
[./SMP]
full = true
type = SMP
petsc_options = '-snes_mf_operator'
[./]
[./FDP_PJFNK]
petsc_options_iname = '-mat_fd_type'
full = true
type = FDP
petsc_options_value = ds
petsc_options = '-snes_mf_operator -
pc_factor_shift_nonzero'
[./]
[./FDP_Newton]
petsc_options_iname = '-mat_fd_type'
full = true
type = FDP
petsc_options_value = ds
petsc_options = '-snes'
[./]
[]
[Executioner]
nl_abs_tol = 1e-8
restart_file_base =
TMI_test_PRA_transient_less_w_ss_out_restart_0831
nl_rel_tol = 1e-5
ss_check_tol = 1e-05
perf_log = true
nl_max_its = 120
type = RavenExecutioner
max_increase = 3
petsc_options_value = lu # '300'
l_max_its = 100 # of linear iterations for each Krylov solve
start_time = 100.0
predictor_scale = 0.6
dtmax = 9999
nl_rel_step_tol = 1e-3
dt = 5e-5
petsc_options_iname = '-pc_type'
e_tol = 10.0
l_tol = 1e-5 # Relative linear tolerance for each Krylov
solve
end_time = 2500.0
e_max = 99999.
[./TimeStepper]
type = FunctionDT
time_t = '0 1. 61.1 100.8 101.5 102.2 120.0 2501.23 1.0e5'
time_dt = '1.e-1 0.40 0.45 0.09 0.1 0.008 0.2 0.21 0.2 0.6'
[./]
[./Quadrature]
type = TRAP
order = FIRST
[./]
[]
[Output]

```

```

output_initial = true
output_displaced = true
file_base = TMI_test_PRA_transient_less_w_out
exodus = true
postprocessor_csv = true
max_pps_rows_screen = 25
[]
[Controlled]
control_logic_input = TMI_PRA_trans_MC_control
[./power_CH1]
print_csv = true
data_type = double
property_name = FUEL:power_fraction
component_name = CH1
[./]
[./power_CH2]
print_csv = true
data_type = double
property_name = FUEL:power_fraction
component_name = CH2
[./]
[./power_CH3]
print_csv = true
data_type = double
property_name = FUEL:power_fraction
component_name = CH3
[./]
[./MassFlowRateIn_SC_A]
print_csv = true
data_type = double
property_name = v_bc
component_name = MassFlowRateIn-SC-A
[./]
[./MassFlowRateIn_SC_B]
print_csv = true
data_type = double
property_name = v_bc
component_name = MassFlowRateIn-SC-B
[./]
[./Head_PumpB]
print_csv = true
data_type = double
property_name = Head
component_name = Pump-B
[./]
[./Head_PumpA]
print_csv = true
data_type = double
property_name = Head
component_name = Pump-A
[./]
[./friction1_SC_A]
print_csv = false
data_type = double
property_name = f
component_name = pipe1-SC-A
[./]
[./friction2_SC_A]
print_csv = false
data_type = double
property_name = f
component_name = pipe2-SC-A
[./]
[./friction1_SC_B]
print_csv = false
data_type = double
property_name = f
component_name = pipe1-SC-B
[./]
[./friction2_SC_B]

```

```

print_csv = false
data_type = double
property_name = f
component_name = pipe2-SC-B
[./]
[/friction1_CL_B]
print_csv = false
data_type = double
property_name = f
component_name = pipe1-CL-B
[./]
[/friction2_CL_B]
print_csv = false
data_type = double
property_name = f
component_name = pipe2-CL-B
[./]
[/friction1_CL_A]
print_csv = false
data_type = double
property_name = f
component_name = pipe1-CL-A
[./]
[/friction2_CL_A]
print_csv = false
data_type = double
property_name = f
component_name = pipe2-CL-A
[./]
[]
[Monitored]
[/avg_temp_clad_CH1]
operator = ElementAverageValue
path = CLAD:TEMPERATURE
data_type = double
component_name = CH1
[./]
[/avg_temp_clad_CH2]
operator = ElementAverageValue
path = CLAD:TEMPERATURE
data_type = double
component_name = CH2
[./]
[/avg_temp_clad_CH3]
operator = ElementAverageValue
path = CLAD:TEMPERATURE
data_type = double
component_name = CH3
[./]
[/avg_Fluid_Vel_H_L-A]
operator = ElementAverageValue
path = VELOCITY
data_type = double
component_name = pipe1-HL-A
[./]
[/avg_Fluid_Vel_C_L_A]
operator = ElementAverageValue
path = VELOCITY
data_type = double
component_name = DownComer-A
[./]
[/avg_out_temp_sec_A]
operator = ElementAverageValue
path = TEMPERATURE
data_type = double
component_name = pipe2-SC-A
[./]
[/DownStreamSpeed]
operator = ElementAverageValue
path = VELOCITY
data_type = double
component_name = pipe1-CL-B
[./]
[/UpstreamSpeed]
operator = ElementAverageValue
path = VELOCITY
data_type = double
component_name = pipe1-CL-B
[./]
[/avg_temp_fuel_CH1]
operator = ElementAverageValue
path = FUEL:TEMPERATURE
data_type = double
component_name = CH1
[./]
[/avg_temp_fuel_CH2]
operator = ElementAverageValue
path = FUEL:TEMPERATURE
data_type = double
component_name = CH2
[./]
[/avg_temp_fuel_CH3]
operator = ElementAverageValue
path = FUEL:TEMPERATURE
data_type = double
component_name = CH3
[./]
[/sec_inlet_velocity]
operator = ElementAverageValue
path = VELOCITY
data_type = double
component_name = pipe1-SC-A
[./]
[]
[Distributions]
RNG_seed = 1
[/crew1DG1]
type = NormalDistribution
mu = 800
sigma = 200
xMin = 0.0
xMax = 2500
[./]
[/crew1DG2CoupledDG1]
type = UniformDistribution
xMin = 0.5
xMax = 1
[./]
[/crewSecPG]
type = NormalDistribution
mu = 1400
sigma = 400
[./]
[/PrimPGrecovery]
type = NormalDistribution
mu = 2000
sigma = 500
[./]
[/CladFailureDist]
type = TriangularDistribution
xMin = 1255.3722
xPeak = 1477.59
xMax = 1699.8167
truncation = 1
lowerBound = 1255.3722
upperBound = 1699.8167
[./]
[]
[RavenAuxiliary]
[/init_exp_frict]

```

```
print_csv = false
data_type = bool
initial_value = True
[./]
[/frict_m]
print_csv = false
data_type = double
initial_value = -1005.56
[./]
[/frict_q]
print_csv = false
data_type = double
initial_value = 10005.1
[./]
[/scram_start_time]
print_csv = true
data_type = double
initial_value = 101.0
[./]
[/friction_time_start_exp]
print_csv = false
data_type = double
initial_value = 0.0
[./]
[/InitialMassFlowPrimary]
print_csv = true
data_type = double
initial_value = 0
[./]
[/initialInletSecPress]
print_csv = false
data_type = double
initial_value = 15219000
[./]
[/CladDamaged]
print_csv = true
data_type = bool
initial_value = False
[./]
[/DeltaTimeScramToAux]
print_csv = true
data_type = double
initial_value = 200.0
[./]
[/InitialOutletSecPress]
print_csv = false
data_type = double
initial_value = 151.7e5 # 15170000
[./]
[/CladTempTreshold]
print_csv = true
data_type = double
initial_value = 1477.59
[./]
[/ScramStatus]
print_csv = true
data_type = bool
initial_value = false
[./]
[/AuxSystemUp]
print_csv = true
data_type = bool
initial_value = false
[./]
[/init_Power_Fraction_CH1]
print_csv = true
data_type = double
initial_value = 3.33672612e-1
[./]
[/init_Power_Fraction_CH2]
```

```
print_csv = true
data_type = double
initial_value = 3.69921461e-1
[./]
[/init_Power_Fraction_CH3]
print_csv = true
data_type = double
initial_value = 2.96405926e-1
[./]
[/a_power_CH1]
print_csv = true
data_type = double
initial_value = 3.33672612e-1
[./]
[/a_power_CH2]
print_csv = true
data_type = double
initial_value = 3.69921461e-1
[./]
[/a_power_CH3]
print_csv = true
data_type = double
initial_value = 2.96405926e-1
[./]
[/a_MassFlowRateIn_SC_A]
print_csv = true
data_type = double
initial_value = 2.542
[./]
[/a_MassFlowRateIn_SC_B]
print_csv = true
data_type = double
initial_value = 2.542
[./]
[/a_Head_PumpB]
print_csv = true
data_type = double
initial_value = 8.9
[./]
[/a_Head_PumpA]
print_csv = true
data_type = double
initial_value = 8.9
[./]
[/a_friction1_SC_A]
print_csv = false
data_type = double
initial_value = 0.001
[./]
[/a_friction2_SC_A]
print_csv = false
data_type = double
initial_value = 0.001
[./]
[/a_friction1_SC_B]
print_csv = false
data_type = double
initial_value = 0.001
[./]
[/a_friction2_SC_B]
print_csv = false
data_type = double
initial_value = 0.001
[./]
[/a_friction1_CL_B]
print_csv = false
data_type = double
initial_value = 0.001
[./]
[/a_friction2_CL_B]
```

```

print_csv = true
data_type = double
initial_value = 0.001
[./]
[/a_friction1_CL_A]
print_csv = false
data_type = double
initial_value = 0.001
[./]
[/a_friction2_CL_A]
print_csv = true
data_type = double
initial_value = 0.001
[./]
[/CladTempTresholdRNG]
print_csv = true
data_type = double
initial_value = 0
[./]
[/auxAbsolute]
print_csv = true
data_type = double
initial_value = 0.001
[./]
[/DG1recoveryTime]
data_type = double
print_csv = true
initial_value = 900
[./]
[/DG2recoveryTime]
data_type = double
print_csv = true
initial_value = 675
[./]
[/SecPGrecoveryTime]
data_type = double
print_csv = true
initial_value = 1800
[./]
[/PrimPGrecoveryTime]
data_type = double
print_csv = true
initial_value = 2000
[./]
[]
[TimeController]
[/cntrAux]
comparisonID = auxAbsolute
time_step_size = 0.01
referenceID = time
delta = 0.5
[./]
[]
[RavenTools]
[/PumpCoastDown]
type = pumpCoastdownExponential
coefficient = 26.5
initial_flow_rate = 8.9
[./]
[/DecayHeatScalingFactor]
type = decayHeat
eq_type = 1
initial_pow = 1
operating_time = 20736000
power_coefficient = 0.74
[./]
[/PumpCoastDownSec]
type = pumpCoastdownExponential
coefficient = 10.5
initial_flow_rate = 1.0

```

6.1.2 Control Logic Input

```
import sys
import math
import distribution1D
import raventools
# initialize distribution container
distcont = distribution1D.DistributionContainer.Instance()
toolcont = raventools.RavenToolsContainer.Instance()

def control_function(monitored, controlled, auxiliary):
    auxiliary.DeltaTimeScramToAux =
    min(auxiliary.DG1recoveryTime+auxiliary.DG2recoveryTime , auxiliary.SecPGrecoveryTime, auxiliary.PrimPGrecoveryTime)

    auxiliary.auxAbsolute = auxiliary.scram_start_time+auxiliary.DeltaTimeScramToAux

    if monitored.time>=(auxiliary.scram_start_time+auxiliary.DeltaTimeScramToAux) and auxiliary.ScamStatus:
        auxiliary.AuxSystemUp = True
    if (monitored.avg_temp_clad_CH1>auxiliary.CladTempTreshold) or (monitored.avg_temp_clad_CH2>auxiliary.CladTempTreshold) or
    (monitored.avg_temp_clad_CH3>auxiliary.CladTempTreshold):
        auxiliary.CladDamaged = True
        auxiliary.a_power_CH1 = controlled.power_CH1
        auxiliary.a_power_CH2 = controlled.power_CH2
        auxiliary.a_power_CH3 = controlled.power_CH3
        auxiliary.a_friction2_CL_B = controlled.friction2_CL_B
        auxiliary.a_friction1_CL_B = controlled.friction1_CL_B
        auxiliary.a_friction2_SC_B = controlled.friction2_SC_B
        auxiliary.a_friction1_SC_B = controlled.friction1_SC_B
        auxiliary.a_friction2_CL_A = controlled.friction2_CL_A
        auxiliary.a_friction1_CL_A = controlled.friction1_CL_A
        auxiliary.a_friction2_SC_A = controlled.friction2_SC_A
        auxiliary.a_friction1_SC_A = controlled.friction1_SC_A
        auxiliary.a_Head_PumpB = controlled.Head_PumpB
        auxiliary.a_Head_PumpA = controlled.Head_PumpA
        auxiliary.a_MassFlowRateIn_SC_B = controlled.MassFlowRateIn_SC_B
        auxiliary.a_MassFlowRateIn_SC_A = controlled.MassFlowRateIn_SC_A

    if monitored.time>=auxiliary.scram_start_time:
        auxiliary.ScamStatus = True
        print('SCRAM')
    else:
        auxiliary.ScamStatus = False
        print('OPERATIONAL STATE')
    #
    if auxiliary.ScamStatus: #we are in scram
        #primary pump B
        if auxiliary.a_Head_PumpB>1.e-4*8.9:
            if not auxiliary.AuxSystemUp: # not yet auxiliary system up
                auxiliary.a_Head_PumpB = toolcont.compute('PumpCoastDown',monitored.time-auxiliary.scram_start_time)
            if auxiliary.a_Head_PumpB < (1.e-4*8.9):
                auxiliary.a_Head_PumpB = 1.e-4*8.9
            auxiliary.a_friction1_SC_B = auxiliary.frict_m*auxiliary.a_Head_PumpB + auxiliary.frict_q
            auxiliary.a_friction2_SC_B = auxiliary.frict_m*auxiliary.a_Head_PumpB + auxiliary.frict_q
            auxiliary.a_friction1_CL_B = auxiliary.frict_m*auxiliary.a_Head_PumpB + auxiliary.frict_q
            auxiliary.a_friction2_CL_B = auxiliary.frict_m*auxiliary.a_Head_PumpB + auxiliary.frict_q
        else: #system up
            if auxiliary.init_exp_frict:
                auxiliary.friction_time_start_exp = auxiliary.a_friction1_SC_B
                auxiliary.init_exp_frict = False
            if auxiliary.a_Head_PumpB <= 0.05*8.9:
                auxiliary.a_Head_PumpB = auxiliary.a_Head_PumpB*1.5
            if auxiliary.a_Head_PumpB > 0.05*8.9:
                auxiliary.a_Head_PumpB = 0.05*8.9
            if auxiliary.a_friction1_SC_B > 0.1:
                auxiliary.a_friction1_SC_B = auxiliary.friction_time_start_exp*math.exp(-(monitored.time-
                (auxiliary.scram_start_time++100.0))/4.0)
                auxiliary.a_friction2_SC_B = auxiliary.a_friction1_SC_B
```

```

    auxiliary.a_friction1_CL_B = auxiliary.a_friction1_SC_B
    auxiliary.a_friction2_CL_B = auxiliary.a_friction1_SC_B
else:
    auxiliary.a_friction1_SC_B = 0.1
    auxiliary.a_friction2_SC_B = 0.1
    auxiliary.a_friction1_CL_B = 0.1
    auxiliary.a_friction2_CL_B = 0.1
else:
    auxiliary.a_Head_PumpB = toolcont.compute('PumpCoastDown',monitored.time-auxiliary.scram_start_time)
    if auxiliary.a_Head_PumpB < (1.e-4*8.9):
        auxiliary.a_Head_PumpB = 1.e-4*8.9
    if auxiliary.a_friction1_SC_B > 0.1:
        auxiliary.a_friction1_SC_B = auxiliary.friction_time_start_exp*math.exp(-(monitored.time-
(auxiliary.scram_start_time++100.0))/4.0)
        auxiliary.a_friction2_SC_B = auxiliary.a_friction1_SC_B
        auxiliary.a_friction1_CL_B = auxiliary.a_friction1_SC_B
        auxiliary.a_friction2_CL_B = auxiliary.a_friction1_SC_B
    else:
        auxiliary.a_friction1_SC_B = 0.1
        auxiliary.a_friction2_SC_B = 0.1
        auxiliary.a_friction1_CL_B = 0.1
        auxiliary.a_friction2_CL_B = 0.1
else:
    if not auxiliary.AuxSystemUp: # not yet auxiliary system up
        auxiliary.a_Head_PumpB = 1.e-4*8.9
        auxiliary.a_friction1_SC_B = 15000
        auxiliary.a_friction2_SC_B = 15000
        auxiliary.a_friction1_CL_B = 15000
        auxiliary.a_friction2_CL_B = 15000
    else:
        if auxiliary.init_exp_frict:
            auxiliary.friction_time_start_exp = auxiliary.a_friction1_SC_B
            auxiliary.init_exp_frict = False
        if auxiliary.a_Head_PumpB <= 0.05*8.9:
            auxiliary.a_Head_PumpB = auxiliary.a_Head_PumpB*1.5
        if auxiliary.a_Head_PumpB > 0.05*8.9:
            auxiliary.a_Head_PumpB = 0.05*8.9
        if auxiliary.a_friction1_SC_B > 0.1:
            auxiliary.a_friction1_SC_B = auxiliary.friction_time_start_exp*math.exp(-(monitored.time-
(auxiliary.scram_start_time++100.0))/4.0)
            auxiliary.a_friction2_SC_B = auxiliary.a_friction1_SC_B
            auxiliary.a_friction1_CL_B = auxiliary.a_friction1_SC_B
            auxiliary.a_friction2_CL_B = auxiliary.a_friction1_SC_B
        else:
            auxiliary.a_friction1_SC_B = 0.1
            auxiliary.a_friction2_SC_B = 0.1
            auxiliary.a_friction1_CL_B = 0.1
            auxiliary.a_friction2_CL_B = 0.1
    else:
        auxiliary.a_Head_PumpB = toolcont.compute('PumpCoastDown',monitored.time-auxiliary.scram_start_time)
        auxiliary.a_friction1_SC_B = auxiliary.frict_m*auxiliary.a_Head_PumpB + auxiliary.frict_q
        auxiliary.a_friction2_SC_B = auxiliary.frict_m*auxiliary.a_Head_PumpB + auxiliary.frict_q
        auxiliary.a_friction1_CL_B = auxiliary.frict_m*auxiliary.a_Head_PumpB + auxiliary.frict_q
        auxiliary.a_friction2_CL_B = auxiliary.frict_m*auxiliary.a_Head_PumpB + auxiliary.frict_q
#primary pump A
auxiliary.a_Head_PumpA = auxiliary.a_Head_PumpB
auxiliary.a_friction1_SC_A = auxiliary.a_friction1_SC_B
auxiliary.a_friction2_SC_A = auxiliary.a_friction2_SC_B
auxiliary.a_friction1_CL_A = auxiliary.a_friction1_CL_B
auxiliary.a_friction2_CL_A = auxiliary.a_friction2_CL_B

#core power following decay heat curve
auxiliary.a_power_CH1 = auxiliary.init_Power_Fraction_CH1*toolcont.compute('DecayHeatScalingFactor',monitored.time-
auxiliary.scram_start_time)
auxiliary.a_power_CH2 = auxiliary.init_Power_Fraction_CH2*toolcont.compute('DecayHeatScalingFactor',monitored.time-
auxiliary.scram_start_time)
auxiliary.a_power_CH3 = auxiliary.init_Power_Fraction_CH3*toolcont.compute('DecayHeatScalingFactor',monitored.time-
auxiliary.scram_start_time)
#secondary system replaced by auxiliary secondary system

```

```

if not auxiliary.AuxSystemUp and auxiliary.ScramStatus: # not yet auxiliary system up
    print('not yet auxiliary system up')
    auxiliary.a_MassFlowRateIn_SC_B = 2.542*toolcont.compute('PumpCoastDownSec',monitored.time-auxiliary.scram_start_time)
    auxiliary.a_MassFlowRateIn_SC_A = 2.542*toolcont.compute('PumpCoastDownSec',monitored.time-auxiliary.scram_start_time)
    if auxiliary.a_MassFlowRateIn_SC_A < (1.e-4*2.542):
        auxiliary.a_MassFlowRateIn_SC_A = 1.e-4*2.542
        auxiliary.a_MassFlowRateIn_SC_B = 1.e-4*2.542
if auxiliary.AuxSystemUp and auxiliary.ScramStatus: # auxiliary system up
    print('auxiliary system up')
    auxiliary.a_MassFlowRateIn_SC_B = auxiliary.a_MassFlowRateIn_SC_B*1.5
    auxiliary.a_MassFlowRateIn_SC_A = auxiliary.a_MassFlowRateIn_SC_B
    if auxiliary.a_MassFlowRateIn_SC_B > 2.542*0.05:
        auxiliary.a_MassFlowRateIn_SC_B = 2.542*0.05
        auxiliary.a_MassFlowRateIn_SC_A = 2.542*0.05
# we work on auxiliaries and we store them back into controlleds
controlled.power_CH1 = auxiliary.a_power_CH1
controlled.power_CH2 = auxiliary.a_power_CH2
controlled.power_CH3 = auxiliary.a_power_CH3
controlled.friction2_CL_B = auxiliary.a_friction2_CL_B
controlled.friction1_CL_B = auxiliary.a_friction1_CL_B
controlled.friction2_SC_B = auxiliary.a_friction2_SC_B
controlled.friction1_SC_B = auxiliary.a_friction1_SC_B
controlled.friction2_CL_A = auxiliary.a_friction2_CL_A
controlled.friction1_CL_A = auxiliary.a_friction1_CL_A
controlled.friction2_SC_A = auxiliary.a_friction2_SC_A
controlled.friction1_SC_A = auxiliary.a_friction1_SC_A
controlled.Head_PumpB = auxiliary.a_Head_PumpB
controlled.Head_PumpA = auxiliary.a_Head_PumpA
controlled.MassFlowRateIn_SC_B = auxiliary.a_MassFlowRateIn_SC_B
controlled.MassFlowRateIn_SC_A = auxiliary.a_MassFlowRateIn_SC_A
if auxiliary.CladDamaged:
    raise NameError ('exit condition reached - failure of the clad')
return

```