

RAVEN and Dynamic Probabilistic Risk Assessment: Software Review

ESREL

Andrea Alfonsi
Cristian Rabiti
Diego Mandelli
Joshua Cogliati
Robert Kinoshita
Antono Naviglio

September 2014

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



This is a preprint of a paper intended for publication in a journal or proceedings. Since changes may be made before publication, this preprint should not be cited or reproduced without permission of the author. This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, or any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for any third party's use, or the results of such use, of any information, apparatus, product or process disclosed in this report, or represents that its use by such third party would not infringe privately owned rights. The views expressed in this paper are not necessarily those of the United States Government or the sponsoring agency.

RAVEN and Dynamic Probabilistic Risk Assessment:

Software Overview

Andrea Alfonsi

Idaho National Laboratory, Idaho Falls, USA. E-mail: andrea.alfonsi@inl.gov.

Cristian Rabiti, Diego Mandelli, Joshua Cogliati, Robert Kinoshita

Idaho National Laboratory, Idaho Falls, USA

Antonio Naviglio

University “La Sapienza”, Rome, Italy

ABSTRACT: Risk Analysis Virtual Environment (RAVEN) is a generic software framework to perform parametric and probabilistic analysis based on the response of complex system codes. The initial development was aimed at providing dynamic risk analysis capabilities to the Reactor Excursion and Leak Analysis Program v.7 (RELAP-7) Thermo-Hydraulic code, currently under development at the Idaho National Laboratory (INL). Although the initial goal has been fully accomplished, RAVEN is now a multi-purpose probabilistic and uncertainty quantification platform, capable to agnostically communicate with any system code. This agnosticism includes providing Application Programming Interfaces (APIs). These APIs are used to allow RAVEN to interact with any code as long as all the parameters that need to be perturbed are accessible through input files or via python interfaces. RAVEN is capable of investigating the system response as well as the input space using Monte Carlo, Grid, or Latin Hyper Cube sampling schemes, but its strength is focused toward system feature discovery, such as limit surfaces, separating regions of the input space leading to system failure, using dynamic supervised learning techniques. The paper presents an overview of the software capabilities and their implementation schemes followed by some application examples.

1 INTRODUCTION

1.1 *Initial Development*

The development of RAVEN (Alfonsi, Rabiti, Mandelli, Cogliati, & Kinoshita 2013) started in 2012, when, within the Nuclear Energy Advanced Modeling and Simulation (NEAMS) program, the need to provide a modern risk evaluation framework became stronger. RAVEN’s principal assignment is to provide the necessary software and algorithms in order to employ the concept developed by the Risk Informed Safety Margin Characterization (RISMC) (Smith, Rabiti, et al. 2012) program. RISMC is one of the pathways defined within the Light Water Reactor Sustainability (LWRS) program. In the RISMC approach, the goal is not just specifically identifying the frequency of an event potentially leading to a system failure, but the closeness (or not) to key safety-related events. This approach may be used in identifying and increasing the safety margins related to those events. A safety margin is a numerical value quantifying the probability that a safety metric (e.g. as peak pressure in a pipe) is exceeded under certain conditions.

The initial development of RAVEN has been focused on providing dynamic risk assessment capability to RELAP-7, currently under development at the INL and, likely, future replacement of the RELAP5-3D (RELAP5-3D Team, 2009) code.

Most of the capabilities implemented having RELAP-7 as principal focus are easily deployable for other system codes. For this reason, several side activities are currently ongoing for coupling RAVEN with software such as BISON, fuel performance, (Williamson, Hales, et al. 2012), RELAP5-3D. This paper is focused on the RAVEN software infrastructure and its functionality.

2 SOFTWARE INFRASTRUCTURE

2.1 *Outlines*

RAVEN has been developed in a highly modular and pluggable way in order to enable easy integration of different programming languages (i.e., C++, Python) and, as already mentioned, coupling with any system code.

RAVEN is composed of three main software systems that can operate either in coupled or stand-alone mode:

- Control Logic System
- Graphical User Interface
- Probabilistic and Parametric framework

The control logic system and the Graphical User Interface are currently available for RELAP-7 only. For this reason, attention is focused on the probabilistic and parametric framework.

2.2 Probabilistic and Parametric framework

The probabilistic and parametric framework represents the core of the RAVEN analysis capabilities. The main idea behind the design of the system is the creation of a multi-purpose framework characterized by high flexibility with respect to the possible performable analysis. The framework must be capable of constructing the analysis/calculation flow at run-time, interpreting the user-defined instructions and assembling the different analysis tasks following a user specified scheme.

In order to achieve such flexibility, combined with reasonably fast development, a programming language naturally suitable for this kind of approach was needed: Python.

Hence, RAVEN is coded in Python and is characterized by an object-oriented design. The core of the analysis performable through RAVEN is represented by a set of basic components (objects) the user can combine, in order to create a custom analysis flow. A list of these components and a summary of their most important functionalities are reported as follows:

- *Distribution*: In order to explore the input/output space, RAVEN requires the capability to perturb the input space (initial conditions of a system code). The initial conditions, that represent the uncertain space, are generally characterized by probability distribution functions (PDFs), which need to be considered when a perturbation is applied. In this respect, a large library of PDFs is available.
- *Sampler*: A proper approach to sample the input space is fundamental for the optimization of the computational time. In RAVEN, a “sampler” employs a unique perturbation strategy that is applied to the input space of a system. The input space is defined through the connection of uncertain variables and their relative probability distributions.
- *Model*: A model is the representation of a physical system (e.g. Nuclear Power Plant); it is therefore capable of predicting the evolution of a system given a coordinate set in the input space.

- *Reduced Order Model (ROM)*: The evaluation of the system response, as a function of the coordinates in the input space, is very computationally expensive, especially when brute-force approaches (e.g. Monte Carlo methods) are chosen as the sampling strategy. ROMs are used to lower this cost, reducing the number of needed points and prioritizing the area of the input space that needs to be explored. They can be considered as an artificial representation of the link between the input and output spaces for a particular system.

The list above is not comprehensive of all the RAVEN framework components (visualization and storage infrastructure).

2.2.1 Distribution

As already mentioned, the perturbation of the input space, through the initial conditions (parameters) affected by uncertainties, needs to be performed by the proper distribution functions. RAVEN provides, through an interface to the BOOST library, the following univariate (truncated and not) distributions: Bernoulli, Binomial, Exponential, Logistic, Lognormal, Normal, Poisson, Triangular, Uniform, Weibull, Gamma, and Beta.

The usage of univariate distributions for sampling initial conditions is based on the assumption that the uncertain parameters are not correlated with each other. Quite often uncertain parameters are subject to correlations and thus the univariate approach is not applicable. This happens when a generic outcome is dependent on different phenomena simultaneously (i.e. the outcome dependency description can not be collapsed to a function of a single variable). RAVEN currently supports both N-dimensional (N-D) PDFs. The user can provide the distribution values on either Cartesian or sparse grid, which determines the interpolation algorithm used in the evaluation of the imported CDF/PDF:

- N-Dimensional Spline (Habermann, & Kindermann 2007), for Cartesian grids
- Inverse weight (Gordon & Wixom 1978), for sparse grids

Internally, RAVEN provides the needed N-D differentiation and integration algorithms to compute the PDF from the CDF and vice versa.

As already mentioned, the sampling methods use the distributions in order to perform probability-weighted perturbations. For example, in the Monte Carlo approach, a random number $\in [0,1]$ is generated (probability threshold) and the CDF, corresponding to that probability, is inverted in order to retrieve the parameter value usable in the simulation. The existence of the inverse for univariate distributions is guaranteed by the monotonicity of the CDF. For N-D distributions this condition is not sufficient since the $CDF(X) \rightarrow$

$[0,1], X \in R^N$ and therefore it could not be a bijective function. From an application point of view, this means the inverse of a N-D CDF is not unique.

As an example, the Figure 1 shows a multivariate normal distribution for a pipe failure as function of the pressure and temperature. The plane identifies an iso-probability surface (in this case, a line) that represents a probability threshold of 50% in this example. Hence, the inverse of this CDF is an infinite number of points.

As easily inferable, the standard sampling approach cannot directly be employed. When multivariate distributions are used, RAVEN implements a surface search algorithm for identifying the iso-probability surface location. Once the location of the surface has been found, RAVEN chooses, randomly, one point on it.

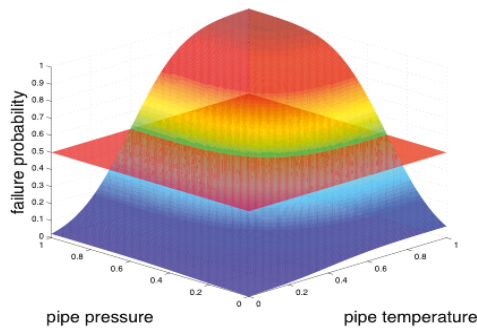


Figure 1. 2-D CDF, function of pressure and temperature

2.2.2 Sampler

The sampler is probably the most important entity in the RAVEN framework. Indeed, it performs the driving of the specific sampling strategy and, hence, determines the effectiveness of the analysis, from both an accuracy and computational point of view. The samplers, that are available in RAVEN, can be categorized in three main classes:

- Once-through
- Dynamic Event Tree (DET)
- Adaptive

Once-through Samplers.

The once-through sampler category collects all the strategies that perform the sampling of the input space without exploiting, through dynamic learning approaches, the information made available from the outcomes of calculation previously performed (adaptive sampling) and the common system evolution (patterns) that different sampled calculations can generate in the phase space (dynamic event tree).

In the RAVEN framework, three different and well-known “once-through” samplers are available:

- Monte Carlo (MC)
- Latin Hyper Cube (LHS)
- Grid Based

As already mentioned, all these sampling strategies are well known, as well as their properties.

Therefore, a detailed investigation of their application is not provided.

Dynamic Event Tree Sampler

In order to clarify the idea behind the DET Sampler currently available in RAVEN, a small overview is needed.

In technological complex systems, as nuclear power plants, an accident scenario begins with an initiating event and then evolves over time through the interaction of dynamics and stochastic events. This mutual action leads to the production of infinitely many different scenarios, which define a continuous dynamic event tree with infinite branches. At each time point, the stochastic variability of the accident outcomes is determined by a multivariate probability distribution. The PRA analysis needs an approximation to this distribution for selected consequence variables. A way to achieve this goal is an Event Tree approach. In dynamic PRA analysis, conventional Event Tree (Alfonsi, Rabiti, Mandelli, Cogliati, Kinoshita, & Naviglio 2013) sequences are run simultaneously starting from a single initiating event. The branches occur at user specified times and/or when an action is required by the operator and/or the system, creating a deterministic sequence of events based on the time of their occurrence. One of the disadvantages of this method is that the timing/sequencing of events and system dynamics is not explicitly accounted for in the analysis. In order to overcome these limitations a “dynamic” approach is needed. The DET (Alfonsi, Rabiti, Mandelli, Cogliati, Kinoshita, & Naviglio 2013) technique brings several advantages, among which is the fact that it simulates probabilistic system evolution in a way that is consistent with the severe accident model. This leads to a more realistic and mechanistically consistent analysis of the system taken into consideration. The dynamic PRA, in general, and the DET methodologies in particular, are designed to take the timing of events explicitly into account, which can become very important especially when uncertain-ties in complex phenomena are considered. Hence, the main idea of this methodology is to let a system code determine the pathway of an accident scenario within a probabilistic “environment”.

From an application point of view, a N-D grid is built on the CDF space. A single simulation is spawned and a set of triggers is added to the system code control logic. Every time a trigger gets activated (one of the CDF thresholds in the grid is overpassed), a new set of simulations (branches) is spawned. Each branch carries its own conditional probability.

As expected, a continuous monitoring of the system evolution in the phase space is needed.

Figure 2 shows a practical example. In this particular case, it is assumed that the probability failure of a pipe depends on the fluid pressure magnitude. Three probability thresholds are defined on the cumulative distribution function. One simulation is spawned (0). As soon as the pressure of the fluid reaches a value corresponding to a 33% probability (CDF), a stop signal is sent and the framework starts two new simulations (branches). The branch in which the system evolved to the newer condition (pipe failed, red line) carries 33% of the probability, while the other the complementary amount. The same procedure is repeated at point 2.

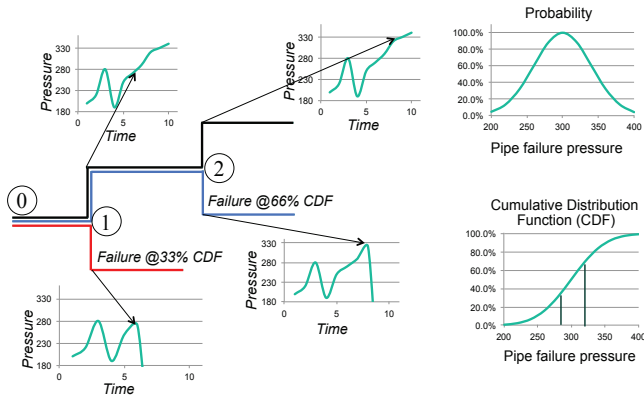


Figure 2. Dynamic Event Tree simulation pattern

Generally, not all the input space can be explored using a DET approach. For instance, usually the parameters affected by aleatory uncertainty are sampled using a dynamic event tree approach, while the ones characterized by epistemic uncertainty are sampled through “once-through” sampling strategies. At the moment a hybrid approach (“once-through” sampling of initial conditions followed by a dynamic event tree strategy) is currently under development.

As already mentioned, this strategy requires a tight interaction between the system code and the sampling driver (i.e., RAVEN framework). In addition, the system code must have a control logic capability (i.e. trigger system). For these reasons, the application of this sampling approach to a generic code needs a bigger effort when compared to the other Samplers available in RAVEN. Currently, the DET is fully available for the thermal-hydraulic codes RELAP-7 and RELAP5-3D.

Adaptive Samplers

A recent feature available within RAVEN is the possibility to perform smart sampling (also known as adaptive sampling) as an alternative to classical “once-through” techniques.

The motivation is that nuclear simulations are often computationally expensive, time-consuming, and high dimensional with respect to the number of

input parameters. Thus, exploring the space of all possible simulation outcomes is infeasible using finite computing resources. During simulation-based probabilistic risk analysis, it is important to discover the relationship between a potentially large number of input parameters and the output of a simulation using as few simulation trials as possible.

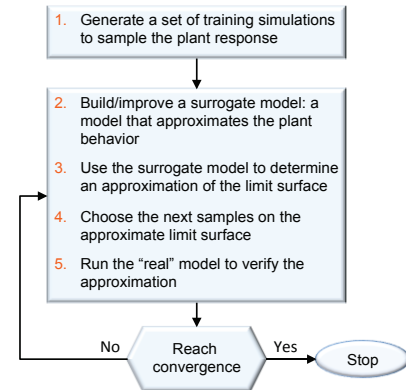


Figure 3. Adaptive Sampling Algorithm

This is a typical context for performing adaptive sampling where a few observations are obtained from the simulation, a reduced order model (ROM) is built to represent the simulation space, and new samples are selected based on the model constructed. The ROM, discussed in section 2.2.4, is then updated based on the simulation results of the sampled points (see Figure 3). In this way, it is attempted to gain the most information possible with a small number of carefully selected sampled points, limiting the number of expensive trials needed to understand features of the system space.

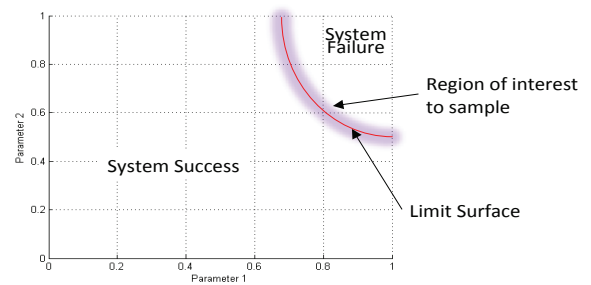


Figure 4. Graphical representation of the Limit Surface

In the following, the specific use case of identifying the limit surface, i.e. the boundaries in the simulation space between system failure and system success (see Figure 4) is analyzed.

For this scope two classes of algorithms are considered:

- Model-based algorithms
- Data-based algorithms

In the first class, the created ROM aims at approximating the real response function of the system as function of the input parameters. Once it is built, it is used to search for the points that are in the proximity of the limit surface using contour reconstruction based algorithms. Response function can be built using Support Vector Machines (SVM)

(Mandelli & Smith 2012) or Kriging based interpolators (Mandelli, Smith, Rabiti, Alfonsi, et al. 2013).

On the other side, data-based algorithms do not build a response function based ROM but determine the location of the limit surface directly from the neighborhood graph constructed from the training data, without any dependencies on a particular prediction model.

These algorithms begin the search of the limit surface by directly building a neighborhood structure as the ROM (e.g. a relaxed Gabriel graph) (Mandelli, Smith, Rabiti, Alfonsi, et al. 2013) on the initial training data. It then creates a candidate set by first obtaining linearly interpolated points along spanning edges (i.e. edge of the graph that connect points having different outcomes) of the graph, and introducing a random perturbation along all dimensions to these points.

2.2.3 Models

The Model entity, in the RAVEN environment, represents a “connection pipeline” between the input and the output space. The RAVEN framework does not own any physical model (i.e. it does not possess the equations needed to simulate a generic system), but implements APIs by which any generic model can be integrated and interrogated. The RAVEN framework provides APIs for three different model categories: Codes, Externals and ROMs.

The Code model represents the communication pipe between the RAVEN and any system code. Currently, RAVEN has APIs for RELAP5-3D, RELAP-7 and any MOOSE (Gaston, Hansen, et al. 2009) based application.

The External model allows the user to create, in a Python file (imported, at run-time, in the RAVEN framework), its own model (e.g. set of equations representing a physical model, connection to another code, control logic, etc.). This model will be interpreted/used by the framework and, at run-time, will become part of RAVEN itself.

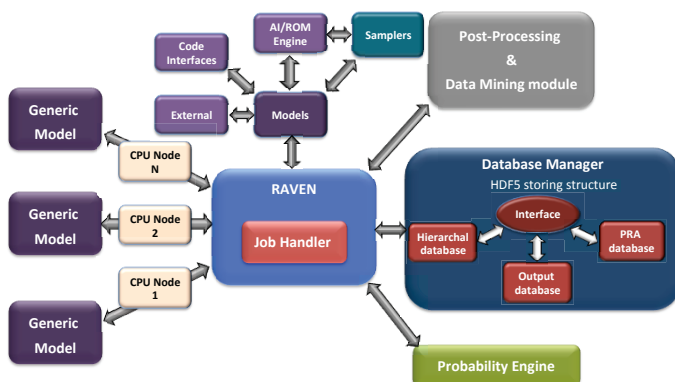


Figure 5. RAVEN probabilistic and parametric framework

The data exchange between RAVEN and the system code can be performed either by direct software interface or by files. If the system code is

parallelized, the data exchanging by files is generally the way to follow since it can be much more optimized in large clusters.

Figure 5 shows a schematic representation of the whole framework, highlighting the communication pipes among the different modules and engines. As can be seen, in the figure all the components discussed so far are reported. In addition the data management, mining and processing modules are shown.

2.2.4 Reduced Order Models

As briefly mentioned, a ROM is a mathematical representation of a system, used to predict a selected output space of a physical system.

The “training” is a process that uses sampling of the physical model to improve the prediction capability (capability to predict the status of the system given a realization of the input space) of the ROM. More specifically, in RAVEN the reduced order model is trained to emulate a high fidelity numerical representation (system codes) of the physical system. Two general characteristics of these models can be generally assumed (even if exceptions are possible):

1. The higher the number of realizations in the training sets, the higher is the accuracy of the prediction performed by the reduced order model. This statement is true for most of the cases although some ROMs might be subject to the over-fitting issues. The over-fitting phenomenon is not discussed here, since its occurrence is highly dependent on the algorithm type, (and there is large number of ROM options available in RAVEN). Every time the user chooses a particular reduced order model algorithm to use, he should consult the relative literature.
2. The smaller the size of the input domain with respect to the variability of the system response, the more likely the surrogate model will be able to represent the system output space.

In most of the cases of interest, the information that is sought is related to defining the failure boundaries of a system with respect to perturbations in the input space. For this reason, in the development of RAVEN, it has been given priority to the introduction of a class of supervised learning algorithms, which are usually referred to as classifiers. A classifier is a reduced order model that is capable of representing the system behavior through a binary response (failure/success).

The first class of classifier introduced has been the SVM (Mandelli & Smith 2012) with several different kernels (polynomial of arbitrary integer order, radial basis function kernel, sigmoid)

followed by a nearest-neighbor based classification using a K-D tree search algorithm. All these supervised learning algorithms have been imported via an API from the scikit-learn (Pedregosa et al. 2011) library. In addition, the N-D spline and the inverse weight methods, that are currently available for the interpolation of N-D PDF/CDF, can also be used as ROMs.

2.2.5 Simulation Environment

RAVEN is perceived by the user as a pool of tools and data. Any action in which the tools are applied to the data is considered a ‘step’ in the RAVEN environment. For the scope of this paper, “multiRun” type of step will be described, since all others are either closely related (single run and adaptive run) or just used to perform data management and visualization. Firstly, the RAVEN input file associates the variable definition syntax to a set of PDFs and to a sampling strategy. The “multiRun” step is used to perform several runs

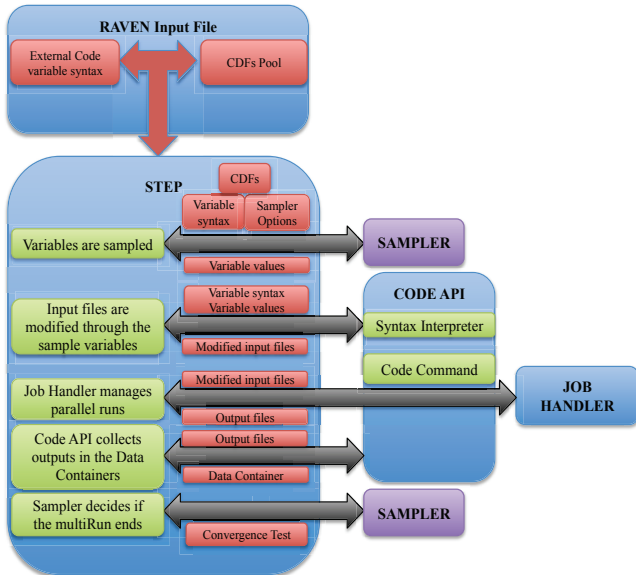


Figure 6. Calculation flow for a Multi-run sampling

(sampling) in a block of a model (e.g. in a MC sampling).

At the beginning of each sub sequential run, the sampler provides the new values of the variables to be modified. The code API places those values properly in the input file. At this point the code API generates the run command and asks to be queued by the job handler. The job handler manages the parallel execution of as many runs as possible within a user prescribed range and communicates with the step controller when a new set of output files are ready to be processed. The code API receives the new input files and collects the data in the RAVEN internal format. The sampler is queried to assess if the sequence of runs is ended, if not, the step controller asks for a new set of values from the

sampler and the sequence is restarted as described in Figure 6.

The job handler is currently capable to run different run instances of the code in parallel and can also handle codes that are multi threaded or using any form of MPI parallel implementation.

RAVEN also has the capability to plot the simulation outcomes while the set of sampling is performed and to store the data for later recovery.

3 EXAMPLES

This section shows examples of analyses performed using RAVEN with three different codes: RELAP-7 (Section 3.1), RELAP5-3D (Section 3.2) and an external model (Section 3.3).

3.1 RELAP-7 analysis

As already mentioned, the first system code that has been driven by RAVEN is RELAP-7. This section shows a station blackout analysis on a simplified Pressurized Water Reactor using RELAP-7 as system code. The plant response has been analyzed using the Monte Carlo and DET approach.

3.1.1 Monte Carlo Analysis

Once the plant is in Station Black-Out (SBO) condition, the probability of recovering the emergency cooling system is set dependent from the recovery of any of the following systems:

- Diesel Generators (DGs): the power is restored when two of the three DG trains are recovered. The recovery time t_{DG1} of the first train is characterized by a normal distribution ($\sigma = 200s$, $\mu = 800s$). The recovery time of the second train t_{DG2} is characterized by a uniform distribution ($\min = 0.5$, $\max=1.0$) and driven by the relation: $\text{pdf}(t_{DG2}) = \text{pdf}(t_{DG1}) \cdot U(x)$.
- The Reserve Station Service Transformer and the main AC line recovery times t_{RSST} and t_{138} are described by normal distributions ($\sigma_{RSST/138} = 500s$, $\mu_{RSST} = 1400s$, $\mu_{138} = 2000s$).

The simulation scope is to assess the failure/success ratio of the plant. The goal function associated is: $\theta_f(T_c MAX, t) = \delta(T - T_{fc})$.

Where $T_c MAX$ and T_{fc} are the maximum and the failure clad temperatures, respectively. The failure temperature of the clad is also a stochastic variable characterized by a triangular distribution ($\min = 1255K$, $\max = 1700K$, $\text{peak} = 1477K$).

Note that the time at which both diesel generators are available needs to be treated as a multidimensional distribution function unless the recovery of the second train is computed inside the

system control logic once the recovery time of the first train is already known. For this analysis, 4000 MC samples were performed.

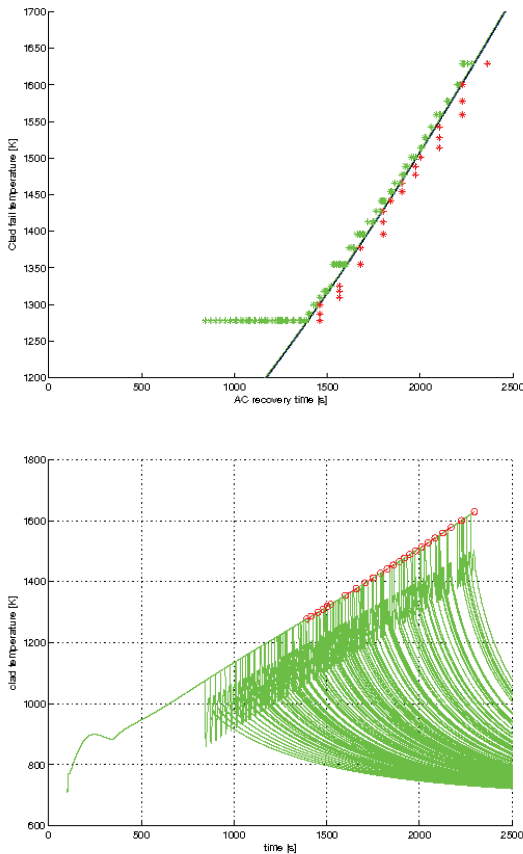


Figure 7. Top: DET a-posteriori Limit Surface. Bottom: Clad temperature evolution

3.1.2 Dynamic Event Tree Analysis

The situation considered is exactly the one presented in the MC analysis (see Section 3.1.1) and the details of the analysis can be found in (Alfonsi, Rabiti, Mandelli, Cogliati, Kinoshita, & Naviglio 2013). Figure 7 shows the time evolution of the clad temperature and a projection of the sampling grid pattern generated by the dynamic event trees approach. The green lines are the simulation continuing while the red dots signal a point where a simulation was stopped when reaching the maximum clad temperature. It could be noticed that there are simulations being stopped at a level of temperature that is exceeded in other simulations. The reason is of course the random value of the clad failure temperature. On the bottom, the plot shows a projection of the threshold triggered by the DET simulation of the transient. The projection is performed by defining the recovery time of the auxiliary system $t_{AC} = \min(t_{DG2}, t_{138}, t_{RSST})$. It is clear how the competing variable (max clad temperature and AC recovery time) are alternatively moved towards higher values of their CDF until a transition point between success and failure is

reached. This pattern is generated by contemporaneously sampling two antagonist variables (or in a terminology more familiar in the RISMIC framework, contemporaneously sampling capacity and load.)

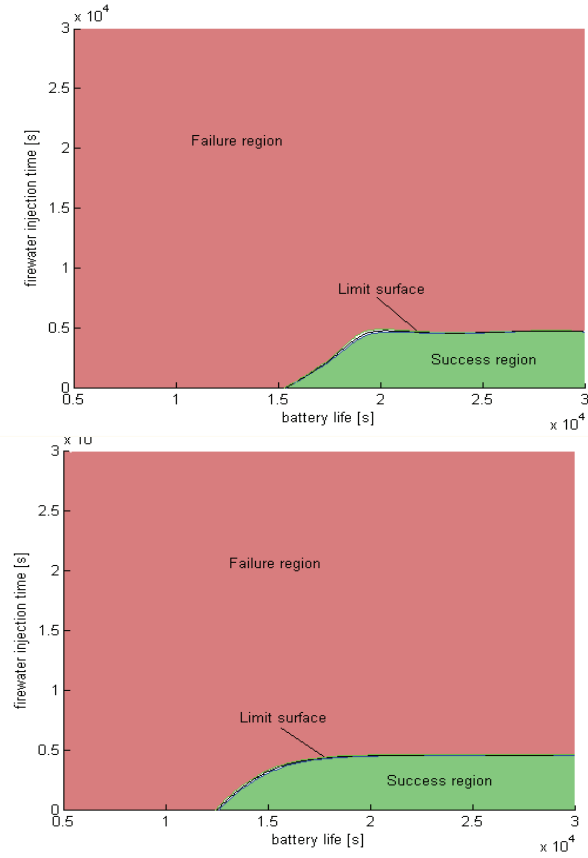


Figure 8. FW availability time vs. Battery life: limit surface for 100% (top) and 120% (bottom) power

3.2 RELAP5-3D BWR SBO analysis

For the RISMIC project, we employed RAVEN and RELAP5-3D to analyze a Station Black-Out (SBO) accident scenario for a Boiling Water Reactor (Mandelli, Smith, Alfonsi, Rabiti, et al. 2014). In particular, we employed both classical statistical tools, i.e. MC, and more advanced machine learning based algorithms to perform uncertainty quantification in order to quantify changes in system performance and limitations as a consequence of power uprate.

In one of these analyses we focused on considering a 2-dimensional state space: firewater (FW) availability time (measured after ADS activation) and battery life. By randomly changing these two parameters we observed the outcome of each simulation (failure or success) and, by using a SVM based classifier, we were able to determine the limit surface.

Results are shown in Figure 8 for two different values of power: 100%, and 120%. As expected, a longer battery life and a shorter firewater injection alignment time lead to success, while a short battery life and long firewater alignment time failed.

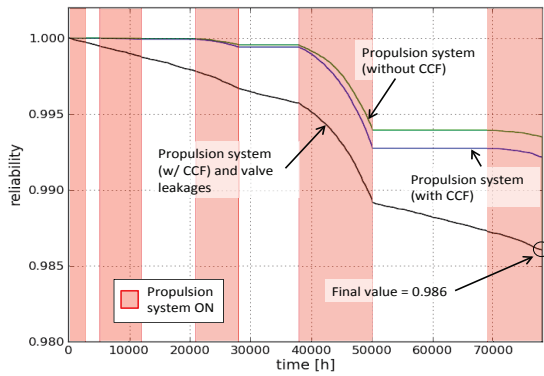


Figure 9. Temporal profile of the PSAM 2014 space propulsion system reliability

3.3 Space Propulsion System

For the PSAM conference, a benchmark problem, a space propulsion system, was issued in order to compare Dynamic PRA methods. In this respect, the RAVEN team proposed a solution to this benchmark problem using RAVEN coupled with an external model that mimics both the dynamic (deterministic) and stochastic (e.g., failure on demands, damage accumulation, common cause failures) behavior of the propulsion system (Mandelli, Smith, Alfonsi & Rabiti 2014). The goal was to measure the temporal profile of the system reliability over the design life (78,000 hours, approximately 9 years). Results are shown in Figure 9 for two different cases: propulsion system only (with and without common cause failures) and propulsion system coupled with valve leakages.

4 CONCLUSIONS

RAVEN is reaching a level of maturity that merits release of the code for testing outside of Idaho National Laboratory. The statistical analysis framework based on grids and Monte Carlo relies on very well assessed methodologies, and has been performing well during internal testing. The integration of those methodologies with the data handling flexibility, the visualization capabilities and the ease of coupling with different physical model simulators show how RAVEN can be a powerful tool for PRA analysis. The DET implementation allows also a rapid turnaround time for the coupling with other codes as long as access to the simulator's control logic is provided. DETs have been identified as one of the most promising approaches for PRA. Thus, the development is currently focused on making the DET an adaptive approach.

Moreover, the coming release will let the PRA community familiarize itself with these enhanced techniques like limit surface or ROM construction

for identifying the leading mechanisms of system failure.

REFERENCES

- Alfonsi, A., Rabiti, C., Mandelli, D., Cogliati, J. & Kinoshita, R. 2013. Raven as a tool for dynamic probabilistic risk assessment: Software overview. *Proceedings of International Conference of mathematics and Computational Methods Applied to Nuclear Science and Engineering*, 1247–1261, San Valley (USA)
- Alfonsi, A., Rabiti, C., Mandelli, D., Cogliati, J., Kinoshita, R., & Naviglio, A. 2013. Dynamic event tree analysis through Raven, In *Proceedings of ANS PSA 2013 International Topical Meeting on Probabilistic Safety Assessment and Analysis*.
- Anders, D., Berry, R. et al., 2012. Relap-7 level 2 milestone report: Demonstration of a steady state single phase PWR simulation with relap-7. Tech. Rep. INL/EXT-12-25924, Idaho National Laboratory.
- Gaston, D., Hansen, G. et al. 2009. Parallel multiphysics algorithms and software for computational nuclear engineering. *Journal of Physics: Conference Series* 180(1).
- Gordon, W.J., & Wixom, J.A. 1978. Shepard's Method of "Metric Interpolation" to Bivariate and Multivariate Interpolation, in *Mathematics and Computation* 32 (141): 253-264.
- Habermann, C. & Kindermann, F. 2007. Multidimensional Spline Interpolation: Theory and Applications. *Computational Economics* 30(2): 153-169.
- Mandelli, D. & Smith C. 2012. Adaptive sampling using support vector machines, in *Proceeding of American Nuclear Society* 107: 736-738.
- Mandelli, D., Smith, C., Rabiti, C., Alfonsi, A., et al. 2013. Dynamic PRA: an overview of new algorithms to generate, analyze and visualize data. In *Proceeding of American Nuclear Society*.
- Mandelli, D., Smith, C., Alfonsi, A., Rabiti, C., et al. 2014. Overview of New Tools to Perform Safety Analysis: BWR Station Black Out Test Case. In *Proceeding of International Conference on Probabilistic Safety Assessment and Management – PSAM*.
- Mandelli, D., Smith, C., Alfonsi, A. & Rabiti, C. 2014. Analysis of the Space Propulsion System Problem Using RAVEN, In *Proceeding of International Conference on Probabilistic Safety Assessment and Management – PSAM*.
- Pedregosa F., et al. 2011. Scikit-learn: Machine Learning in Python”, *Journal of Machine Learning Research*, 2825-2830.
- Rabiti, C., Mandelli, D., et al. 2012. Reactor analysis and virtual control environment (Raven) FY12 report. Tech. Rep. INL/EXT-1227351, Idaho National Laboratory.
- RELAP5-3D Team, 2009. RELAP5-3D Code Manual. Tech. Rep. INEEL-EXT-98-00834, Idaho National Laboratory.
- Smith, C., Rabiti, C., et al. 2012. Risk Informed Safety Margins Characterization (RISMC) Pathway Technical Program Plan. Tech. Rep. INL/EXT-11-22977, Idaho National Laboratory.
- Williamson, R.L., Hales, J.D. et al 2012. Multidimensional multiphysics simulation of nuclear fuel behavior, *Journal of Nuclear Materials* 423:149–163.