INL/EXT-14-32491
Revision: 0

Advanced Probabilistic Risk Analysis Using RAVEN and RELAP-7

Cristian Rabiti
Andrea Alfonsi
Diego Mandelli
Joshua Cogliati
Robert Kinoshita

June 2014

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

CONTENTS

FIGURES

TABLES

1. INTRODUCTION

RAVEN, under the support of the Nuclear Energy Advanced Modeling and Simulation (NEAMS) program [1], is advancing its capability to perform statistical analyses of stochastic dynamic systems. This is aligned with its mission to provide the tools needed by the Risk Informed Safety Margin Characterization (RISMC) path-lead [2] under the Department of Energy (DOE) Light Water Reactor Sustainability program [3]. In particular this task is focused on the synergetic development with the RELAP-7 [4] code to advance the state of the art on the safety analysis of nuclear power plants (NPP).

The investigation of the probabilistic evolution of accident scenarios for a complex system such as a NPP is not a trivial challenge. The complexity of the system to be modeled leads to demanding computational requirements even to simulate one of the many possible evolutions of an accident scenario (tens of CPU/hour). At the same time, the probabilistic analysis requires thousands of runs to investigate outcomes characterized by low probability and severe consequence (tail problem).

The milestone reported in June of 2013 [5] described the capability of RAVEN to implement complex control logic and provide an adequate support for the exploration of the probabilistic space using a Monte Carlo sampling strategy. Unfortunately the Monte Carlo approach is ineffective with problems of such complexity.

In the following year of development, the RAVEN code has been extended with more sophisticated sampling strategies (grids, Latin Hypercube, and adaptive sampling). This milestone illustrates the effectiveness of those methodologies in performing the assessment of the probability of core damage following the onset of a Station Black Out (SBO) situation in a boiling water reactor (BWR).

Section 2 provides an overview of the available probabilistic analysis capabilities, ranging from the different types of distributions available, possible sampling strategies, and post processing analysis capabilities. Section 2 also provides an extensive description of two major developments introduced this year: adaptive sampling for limit surface sampling and multi variate distributions. The document concludes with a description of the demo case (BWR-SBO) and a discussion of the results obtained.

2. PROBABILISTIC ANALYSIS TOOLS

Given the probabilistic aspects characterizing the initiating events of an accident scenario in a NPP and its evolution (e.g. failure on demand), it is natural to try to evaluate the risk (R) connected to the operation of a NPP in a probabilistic sense (referred to as a Probabilistic Risk Analysis [PRA]). Before entering into the detail of the work completed, the recall of a few mathematical concepts will help the understanding of the material.

Specifically:

- $\bar{x} = \bar{X}(t)$: random variate representing the status of the system (e.g., NPP status) at a given instant in time
- $\bar{X}(t)$: random variable. For each possible realization of the stochastic components of the system determine the status of the system; it is thus a mapping between the event space and the system phase space
- $f_{\bar{X}}(\bar{x}, t)$: probability distribution function (PDF) of $\bar{x}$ with support S
- $F_{\bar{X}}(\bar{x}, t) = \int_{\partial S^-}^{<\bar{x}} f(\bar{x}', t) d\bar{x}'$: cumulative distribution function (CDF),
- $C(\bar{x}, t)$: cost functions which represent the maximum risk given the status of the system $\bar{x}$ at time t,

- $R = \int_0^\infty dt \int_S C(x,t) f_{\bar{X}}(\bar{x},t) d\bar{x}$: risk. It is useful to remark that the risk integral is nothing more than the integral over time of the expected value of the cost function $E[C(x)]$.

To illustrate these quantities in a more practical sense, in a prototypical PRA the following correspondence could be used:

- $C(\bar{x}) = \begin{cases} \delta(t_{end}) \; if \; reactor \; is \; damaged \\ 0 \; if \; recator \; is \; not \; damaged \end{cases}$,
- $R = \int_{S \cap reactor \; damaged} f_X(\bar{x}, t_{end}) d\bar{x}$,

where $\delta(.)$ is the Dirac delta and t_{end} is the maximum time extension of the simulation used for the PRA analysis. In such a case the risk becomes just the probability that before the end time of the simulation the reactor has been damaged. As it could be noticed, since the goal function is known, the PRA analysis task is therefore the computation of some form of the $F_X(x, t_{end})$ or more in general of $F_{\bar{X}}(\bar{x}, t)$ that is what is needed for computing the risk integral.

Before describing the details of how this could be performed (this subject is treated in Section 2.3, dedicated to the sampling strategies) the above example could be further extended to make a more detailed parallel between the mathematical representation of the problem insofar provided and the RAVEN-RELAP-7 simulation environment.

When RAVEN and RELAP-7 are used to perform a PRA the following correspondence are assumed:

- The realization space (scenario space) is the set of all possible values that all parameters subject to stochastic behavior, used in the construct of the numerical model solved by RELAP-7, could possible assume. Examples are:
    - Friction coefficient
    - Time of the recovery the auxiliary system
    - Number of successful time a valve is operated before failing.
- A realization (scenario) is characterized by a specific set of values of the above-described random variables.
- RELAP-7 is the mapping (random variable) from the scenario space to the outcome space (random variate).

In the software infrastructure constructed, RAVEN is the scenario generator that—according to the probability distribution function characteristic of each random variable—creates scenarios, monitors the resulting value of the random variate describing the plant status (computed by RELAP-7), infers their probabilistic distribution functions, and compute the risk integrals.

The following subsections summarize the capability of RAVEN to represent stochastic behavior by the implemented probability distribution functions, and describe the different strategies (samplers) to compute the cumulative distribution function of the status of the system ($F_{\bar{X}}$) or the risk integrals.

2.1 Univariate Distributions

Both the probabilistic distribution of events (e.g. pump failure) or the uncertainty characterizing one of the model parameters used to describe the NPP (e.g., friction coefficient) are described by PDFs.

In RAVEN a large number of analytical PDFs have been made available during the current fiscal year. Table 1 summarizes the distributions present and if their truncated form are available. Figure 1

illustrates an example of the probability plots generated by some PDFs. The distributions have been imported via the construction of a software interface with the BOOST library [6]. The truncation of a distribution in an interval [a,b], if available, is performed to preserve the normalization using the following transformation:

$$f_X(x|a \le x \le b) = \frac{f_X(x)}{f_X(b) - f_X(a)}$$

Table 1: List available probability distribution functions

Probability Distribution Function	*Truncated Form Available*	*Probability Distribution Function*	*Truncated Form Available*
Bernoulli	*No*	*Poisson*	*No*
Binomial	*No*	*Triangular*	*Yes*
Exponential	*Yes*	*Uniform*	*Yes*
Logistic	*Yes*	*Weibull*	*Yes*
Lognormal	*Yes*	*Gamma*	*Yes*
Normal	*Yes*	*Beta*	*Yes*

In the following are reported some distribution analytical expressions used to generate the plots in Figure 1 as also used to generate the example of the different sampling strategies in Figure 4 to Figure 10.

$$Normal(x; \sigma = 0.5, \mu = 0) = N(x; \sigma = 0.5, \mu = 0) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

$$Lognormal(x; \sigma = 0.5, \mu = 0) = LN(x; \sigma = 0.6, \mu = 0) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(ln(x)-\mu)^2}{2\sigma^2}},$$

$$Uniform(x; min = 0, max = 3) = U(x; min = 0, max = 3) = \frac{[H(x)-H(3-x)]}{3}.$$

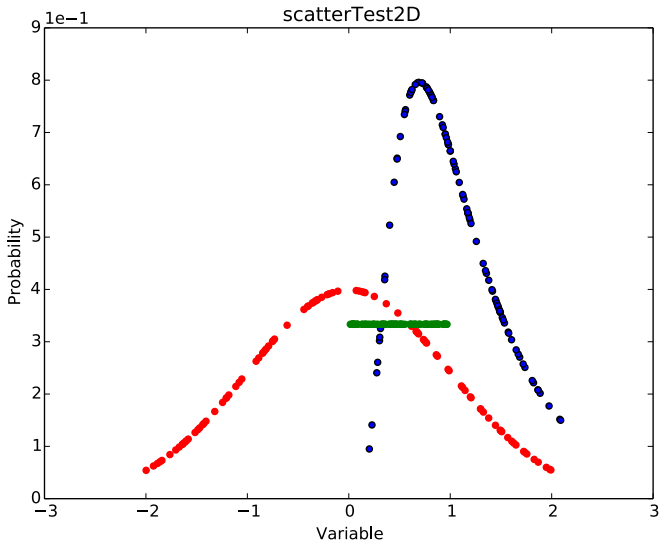

Figure 1: Scattered plot generated by sampling (Latin Hypercube sampling scheme) of a Normal (red), Lognormal (blue) and Uniform distributions (green)

2.2 Multivariate Distributions

A univariate distribution statistically represents the uncertainty associated to a 1-dimensional parameter (e.g., AC power recovery time, pipe friction factor). As an example, a normally distributed (mean μ and sigma σ) variable x can be written as:

$$x \sim \mathcal{N}(\mu, \sigma),$$

and the PDF associated to x can be written as:

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

Such concept can be extended to multi-dimensional, i.e. multivariate, distributions [7] in order to model how the uncertainties associated to n coupled parameters $(x_1, \dots, x_n)$ are distributed and correlate among each other (e.g., concentration of H_2 and CO in the LWR containment at which ignition will start). In such case we are dealing with n-dimensional variable, i.e., a vector $\bar{x} = [x_1, \dots, x_n] \in \mathbb{R}^n$, which has an associated distribution $\mathcal{M}$:

$$\bar{x} \sim \mathcal{M}.$$

Note that concepts like PDFs $f_{\bar{X}}(\bar{x})$:

$$f_{\bar{X}}(\bar{x}): \mathbb{R}^n \to \mathbb{R}, f_{\bar{X}}(\bar{x}) = f_{\bar{X}}(x_1, \dots, x_n),$$

and CDFs $F_{\bar{X}}(\bar{x})$:

$$F_{\bar{X}}(\bar{x}): \mathbb{R}^n \to [0,1], F_{\bar{X}}(\bar{x}) = F_{\bar{X}}(x_1, \dots, x_n),$$

can still be defined similarly to the univariate case, i.e.,

$$F_{\bar{X}}(\bar{x}) = F_{\bar{X}}(x_1, \dots, x_n) = \int_{-\infty}^{x_1} dx_1 \dots \int_{-\infty}^{x_n} dx_n \, f_{\bar{X}}(x_1, \dots, x_n),$$

$$f_{\bar{X}}(\bar{x}) = f_{\bar{X}}(x_1, \dots, x_n) = \frac{\partial^n}{\partial x_1 \dots \partial x_n} F_{\bar{X}}(x_1, \dots, x_n).$$

However, operations like the inverse of the CDF (performed to generate random samples from a given CDF) need to be treated differently. The inverse of a multi-dimensional function is in fact not a uniquely determined point but an infinite set of points distributed on a line (for a 2-dimensional multivariate distribution), or on a surface (for a 3-dimensional multivariate distribution) and so on. In other words, given a value $p \in [0,1]$, $F^{-1}(p)$ determine an infinite set of points lying in a $(n-1)$-dimensional hyper surface constrained by the same value of the CDF that iso-probability surface (see Figure 2).

2.2.1 Normal Multivariate Distributions

Since multivariate distributions are widely used in the UQ and PRA arenas, a set of stochastic libraries that can handle those distributions has been developed for the RAVEN code. In the literature, multivariate normal distributions are mainly considered. Those distributions are defined as [7]:

$$\bar{x} \sim \mathcal{N}(\bar{\mu}, \bar{\bar{\Sigma}})$$

where:

- $\bar{\mu} = [\mu_1, \dots, \mu_n] \in \mathbb{R}^n$ is the mean value,
- $\bar{\bar{\Sigma}}$ is the covariance matrix.

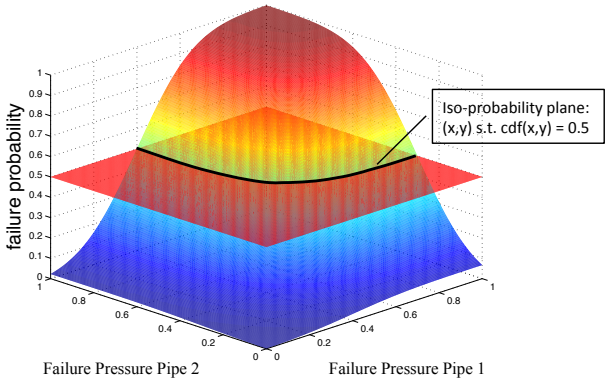

Figure 2: Example of multivariate cdf $F(x, y)$ (2D case): the average internal pressure at which two different pipe will fail. The correlation could be given by being built by the same material. The iso-probability planes represents the location of the points (x,y) such that the $F(x, y)$ =0.5

While an analytical expression for a multivariate normal PDF exists:

$$f_{\bar{X}}(\bar{x}) = f_{\bar{X}}(x_1, \dots, x_n) = \frac{1}{\sqrt{(2\pi)^n \left|\bar{\bar{\Sigma}}\right|}} e^{-\frac{1}{2}(\bar{x}-\bar{\mu})^T \bar{\bar{\Sigma}}^{-1}(\bar{x}-\bar{\mu})},$$

an expression for a multivariate normal CDF $F(x)$ does not exists but can be determined numerically.

As an example, for a 2-dimensional normal distribution, the covariance matrix can be written as:

$$\begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix},$$

where σ_1 and σ_2 are the standard deviations associated to the two variables x_1 and x_2 while ρ represents the correlation factor between x_1 and x_2. If x_1 and x_2 are not correlated, i.e., $\rho = 0$, then:

$$f_{X_1,X_2}(x_1,x_2) = \frac{1}{\sqrt{(2\pi\sigma_1\sigma_2)^2}} e^{-\frac{1}{2}\left[\frac{(x_1-\mu_1)}{\sigma_1^2}+\frac{(x_2-\mu_2)}{\sigma_2^2}\right]} = \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{(x_1-\mu_1)^2}{2\sigma_1^2}} \cdot \frac{1}{\sigma_2\sqrt{2\pi}} e^{-\frac{(x_2-\mu_2)^2}{2\sigma_2^2}}$$
$$= f_{X_1}(x_1) \cdot f_{X_2}(x_2),$$

i.e., the multivariate normal distribution can be written as a product of two independent univariate distributions (for x_1 an x_2).

2.2.2 N-dimensional spline interpolation

The development of multivariate distributions was not limited to model only normal distributions but we also added the capability to handle custom multivariate distributions. In such cases, the user provide through a .txt file a set of point (M) of coordinates $\{\bar{x}\}_i = \{(x_1, \dots, x_n) \in \mathbb{R}^n\}_i, i = 1, \dots, M$ in the *n*-dimensional space and the value of the CDF $F_{\bar{X}}(\bar{x})$ associated to those points. In order to calculate the value of $F_{\bar{X}}(\bar{x})$ at a generic point, a set *n*-dimensional interpolation functions have been employed. This section is dedicated to the description of an interpolating process that uses *n*-dimensional spline while the next one will focus on inverse weight based methodologies.

Given that the data points provided by the user are located on a Cartesian (regular) grid (see Figure 3), RAVEN allows using a cubic spline interpolation. Such interpolation method constructs a representation of the original data set that is both continuous and differentiable.

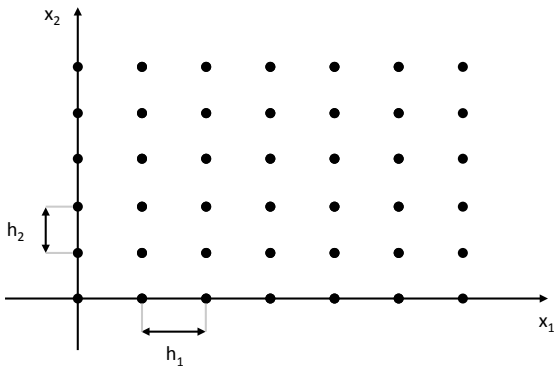

Figure 3: 2-dimensional points lying on a Cartesian (regular) grid

Given a set of M points and CDF values $\{\bar{x}, F_{\bar{X}}(\bar{x})\}_i,\ i = 1, \dots, M$ lying on a n-dimensional Cartesian grid with n different discretization step h_j for $j = 1, \dots, n$ (one for each dimension), the interpolated value $F(\bar{x})$ at a generic point $\bar{x} = [x_1, \dots, x_n] \in \mathbb{R}^n$ can be written as [8]:

$$F_{\bar{X}}(\bar{x}) = \sum_{j_1=1}^{m_1+2} \dots \sum_{j_n=1}^{m_n+2} c_{j_1,j_2,\dots,j_n} \cdot \prod_{k=1}^{n} u_{j_k}^k(x_k)$$

where:

$m_j, j = 1, \dots, n$: number of discretization points for each dimension (j)

$$\prod_{j=1}^{n} m_j = M$$

$$u_{j_k}^k(x_k) = \Phi\left(\frac{x_k - x_k^L}{h_k} + 2 - j_k\right)$$

$$\Phi(t) = \begin{cases} (2-|t|)^3 & 1 \le |t| \le 2 \\ 4 - 6|t|^2 + 3|t|^3 & |t| < 1 \\ 0 & elsewhere \end{cases}$$

The scope of the interpolator is to determine all the coefficients $c_{j_1,j_2,\dots,j_n}$ through a recursive algorithm as shown in [8]. Note that dimensionality of the problem is $\prod_{j=1}^{n}(m_j + 2)$. Hence, for high dimensionality problem and with a large number of discretization points, the calculation of the coefficients $c_{j_1,j_2,\dots,j_n}$ may be computationally intensive.

2.2.3 Inverse weight interpolation

Also know as Shepard's interpolator method [9], the inverse weight algorithm perform n-dimensional interpolation using a basic metric distance scheme.

Given a set of M points and CDF values $\{\bar{x}, F_{\bar{X}}(\bar{x})\}_i,\ i = 1, \dots, M$ the interpolated value $F(\bar{x})$ at a generic point $\bar{x} = [x_1, \dots, x_n] \in \mathbb{R}^n$ can be written:

$$F_{\bar{X}}(\bar{x}) = \sum_{i=1}^{M} w_i(\bar{x})\, F(\bar{x}_i)$$

where:

$$w_i(\bar{x}) = \frac{\left\|\bar{x}_i - \bar{x}_j\right\|^p}{\sum_{j=1}^{M}\left\|\bar{x}_i - \bar{x}_j\right\|^p}$$

with p set to a value greater than 2 in order to assure differentiability.

2.2.4 Challenges

While dealing with univariate distributions is common practice, the multivariate distributions pose a set of challenges that require special handling approaches

2.2.4.1 Computation of the inverse of a multivariate cumulative distribution function:

Given the value of the $F_{\bar{X}}(\bar{x}') = P(\bar{x} \le \bar{x}')$, to compute the corresponding value of the random variate $\bar{x}'$ (if exists) is a problem that occurs frequently when performing this type of statistical analysis. This operation is for example required to generate a set of sampling of a random variable using the inverse transformation method in Monte Carlo based analysis (see Section 2.3.1).

While for univariate distribution the solution of $\bar{x}' = F_{\bar{X}}^{-1}\left(P(\bar{x} \le \bar{x}')\right)$ could be always computed (either analytically or numerically), in the case of multivariate distribution the equation has an infinite number of solutions. However, using an iterative scheme, it is possible to find a point $\bar{x}' \in \mathbb{R}^n$ such that $F_{\bar{X}}(\bar{x}') = P(\bar{x} \le \bar{x}')$ using for example graph-based methods [10]. Unfortunately ensuring that the $\bar{x}'$ not only satisfy the above constrain but also is uniformly distributed over the hyper-surface (iso-probability surface) is rather challenging. Preliminary results unfortunately show that the search algorithm used to locate $\bar{x}$ has effectively some bias. While the effect seems to be negligible, more investigation will be required in the future.

2.2.4.2 Derivation of the probability distribution function from the cumulative distribution function

As indicated earlier the user provides the point evaluation of the $F_{\bar{X}}(\bar{x})$ CDF, therefore the PDF $f_{\bar{X}}(\bar{x})$ should be derived by differentiation:

$$f_{\bar{X}}(\bar{x}) = \frac{\partial^n}{\partial x_1 \dots \partial x_n} F_{\bar{X}}(x_1, \dots, x_n)$$

Such $f_{\bar{X}}(\bar{x})$ is affected by errors generated by both the interpolation method employed and by the quantity of points $\bar{x}_i$ given as input by the user. In this respect, an estimate of such error is needed and should be argument of investigation in the future.

2.2.5 Parametric Distributions

In PRA analysis there are also several examples of parametric distributions. While in many respects similar to the multivariate case, the treatment of those distributions should be different from a probabilistic point of view.

Referring to the notation introduced in Section 1:

- x: random variate
- $X(\bar{p})$: random variable, function of the parameter $\bar{p}$
- $X(\bar{p}) \sim f_X(x, \bar{p})$: PDF
- $F_X(x, \bar{p}) = \int_{x_{min}(\bar{p})}^{x} f(x', \bar{p}) dx$: CDF

Differently form a multivariate distribution:

$$\int_{x_{min}(\bar{p})}^{x_{max}(\bar{p})} f(x', \bar{p}) dx = 1$$

$$\int_{\bar{p}} d\bar{p}' \int_{x_{min}(\bar{p})}^{x_{max}(\bar{p})} f(x', \bar{p}') dx = Vol(\bar{p})$$

It could be noticed as for any value of the parameter $\bar{p}$ the normalization condition is respected and the integration for all possible parameter values leads to the iper-volume $Vol(\bar{p})$ representing the estension of the $\bar{p}$ domain. In reality this situation is not very much different from what has been described in chapter 1 where the natural parameterization chosen for the status of the system is the time coordinate. In fact the probability of the system being in any of the admissible points is always 1 and its integral over time is just the total simulation time. In the analysis presented in the latter part of this report a parametric distribution is used to represent the evolution of the failure clad temperature probability as a function of the burn-up.

2.3 Samplers

As already mentioned a sampler is an algorithm which purpose is to determine the cumulative distribution function of the status of the system or to compute risk integrals.

2.3.1 Sampling of Probability Distribution Function

Before moving forward into the description of the different sampling techniques it is useful to recall some results that show how the sampling of different type of distribution could be performed.

2.3.1.1 Transformation Method

This methodology is used when the inverse of the CDF is know analytically

- if X, Y random variable with their respective random variate x, y, PDF f_X, f_Y, and cdf F_X, F_Y
- Then $y = F_Y^{-1}\big(f_X(x)\big)$ or equivalently $Y = F_Y^{-1}(f_X)$

A special case is when, conveniently $f_X = \mathcal{U}(0,1)$, and this lead to $y = F_Y^{-1}(x)$. This result tells that the realization of any random variable could be obtained by computing the inverse of its CDF on the realization of a random variable uniformly distributed on $[0,1]$.

2.3.1.2 Rejection Method

This methodology is numerically based, and used when the inverse of the CDF is not known analytically. Once a sampling u is generated from the $\mathcal{U}(0,1)$ the solution of $F_Y(y) = u$ is sought iteratively. The process, of course, takes advantage of the monotonicity of the CDF.

2.3.1.3 Multivariate and Parametric Distributions

In case of multivariate CDF clearly the inverse does not exist given that the inversion problem is ill conditioned. As already mentioned in Section 2.2.4 the inversion is performed numerically but the process currently implemented needs further refining to ensure the complete absence of bias in the location of the point on the iso-probable surface.

To focalize the issue it could be helpful to examine the following example. Referring to the case in Figure 2. $(P_{1F}, P_{2F}) \sim f_{T_{1F},T_{2F}}(p_{1F}, p_{2F})$: random variate where, respectively, the two variates are the failure temperature of pipe 1, and failure temperature of pipe 2. The inversion method would prescribe to generate a random number x realization of the uniform distribution $\mathcal{U}(0,1)$ and solve:

$$F_{P_{1F},P_{2F}}(p_{1F}, p_{2F}) = \int_{p_{1F,min}}^{p_{1F}} dp_{1F}{}' \int_{p_{2F,min}}^{p_{2F}} dp_{2F}{}' f_{P_{1F},P_{2F}}(p_{1F}, p_{2F}) = x.$$

Clearly this would lead to the definition of an hyper-surface (line in this case) satisfying the constrain:

$$F_{P_{1F},p_{2F}}(p_{1F}, p_{2F}) = x.$$

While it is possible to numerically determine a couple of value $(\tilde{p}_{1F}, \tilde{p}_{2F})$ that satisfies the above constrain it is rather challenging to define an algorithm that used several times for the same x will produce a set of Independent Identically Distributed (IID) pairs of $(\tilde{p}_{1F}, \tilde{p}_{2F})_i$ as requested for example by Monte Carlo sampling strategy. As mentioned in Section 2.2.4 this issue is still under investigation. Currently RAVEN provides the capability to perform random sampling of multi-dimensional distribution but the IID hypothesis is not fully ensured. Nonetheless the bias and its effect detected, in test cases, so far have been minimal.

2.3.2 Monte Carlo

The Monte Carlo sampling strategy is the most used and more basic sampling strategy. First thing is to remind that the risk integrals are just the expected values of the cost function $E[C(x)]$ therefore using the Law of Large Numbers [11] it is possible to compute an approximation of the risk integrals.

The sequence of steps could be so summarized as follows:

1. M IID realizations $\{\bar{u}_i\}\ i = 1, \dots, M$ are generated. The vectors $\bar{u}_i$ are constitute by *N* realizations of the uniform distribution $\mathcal{U}(0,1)$. Examples are friction coefficient or recovery time of an auxiliary system (monovariate) or the temperature and pressure of failure of a pipe (multivariate).
2. For each vector $\bar{u}_i$ for each entry of the vector $u_{i,j}, j = 1, \dots, N$ (N is the number of random variables, counting the multidimensional ones as one) the corresponding values of the variates $\bar{v}_i$are generated (stochastic parameters used to define the mathematical representation of the system) using either the transformation method, either the rejection method, or numerical inversion for the multivariate. To be notice that $dim\{\bar{u}_i\} \geq dim\{\bar{v}_i\}$ and that only after the inversion of the multivariate distributions the number of parameter generated equals the number of stochastic parameters needed by the RELAP-7 model.

3. For each set $\bar{v}_i$ a RELAP-7 run is executed producing the time evolution of the coordinate in the phase space representing the system (aka NPP) $x_i(t) = h(\bar{v}_i, t)$. Where the transfer function $h(\bar{v}_i, t)$ represent the mathematical model used by RELAP-7 to represent the system.
4. The value of the goal function $C_i(x_i(t))$ is computed.
5. To each C_i it is associated the probability $\frac{1}{M}$ and therefore it is possible to compute its expected value by: $E[C] = \frac{1}{M}\sum_{i=1}^{M} C_i$

Figure 4 shows the sampling points (300 points) resulting from a Monte Carlo approach over three monovariate parameters sampled respectively from uniform (Variable_U), lognormal (Variable_LN), and normal (Variable_N) distributions. Figure 5 shows instead the sampled value of a goal function set equal to the product of the lognormal and normal probability distribution functions. The second image clearly shows the clustering of the sampling location towards the highest probability region, reflecting the characteristic of the Monte Carlo approach to better represent goal functions (risk integrals) close to highest probable regions, while performing a poor job in the assessment of the system behavior in low probability regions.

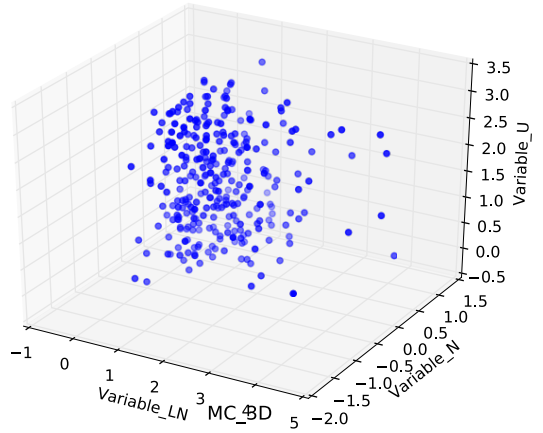

Figure 4: Scattered plot of the location of the sampling point resulting from a Monte Carlo sampling over three monovariate parameters

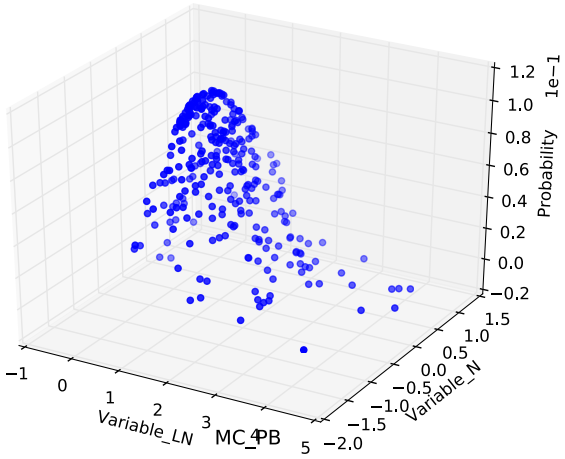

Figure 5: Probability values sampled by the Monte Carlo

2.3.3 Grid Sampling

Grid-based sampling is more aimed towards a parametric analysis of the system response rather than a probabilistic analysis. RAVEN for monovariate parameters allow either defining the grid in terms of probability (CDF) or directly with respect the parameter values. There is the limitation that, when using grid build on the CDF values, the distribution used should have a finite interval support so to avoid that the grid point expressed in parameter values are $\pm\infty$ (this would happen if the support of the distribution is $\pm\infty$ and the point chosen on the grid are 0 or 1). For the parametric distributions it is only possible to define a grid with respect to the CDF since the support of the PDF evolves with the evolution of the system. Also for multivariate distributions it is not possible to define the grid over the values of the parameters but only over the CDF, in fact, similar to the parametric case, the different variable could not discretized separately.

The procedure to evaluate the risk integrals is similar to the Monte Carlo, except that the probability associated to each sampling should be computed differently:

- First if the grid points are provided in values, than they are re-mapped in term of grid point over [0,1] by means of the inverse of the CDF
- $\{\bar{u}_i\}\ i = 1, \dots, M$: i-th point on the grid of coordinate $u_{i,j}, j = 1, \dots, N$, where N is the number of dimension on the grid
- $Pr(\bar{u}_i) = \prod_{j=1}^{N} \left(F_{U_i}(u_{i,j+1/2}) - F_{U_i}(u_{i,j-1/2}) \right)$
- Where the index $j \pm 1/2$ refers to the coordinate located at the middle point between the coordinate $u_{i,j}$ and the next/previous point on the grid.
- For the extreme points it is used the relationship:

  if $F_{U_i}(u_{i,j+1/2}) > F_{U_i}(u_{i,j-1/2})$ then $F_{U_i}(u_{i,N+1/2}) = 1, and\ F_{U_i}(u_{i,1-1/2}) = 0$

  if $F_{U_i}(u_{i,j+1/2}) < F_{U_i}(u_{i,j-1/2})$ then $F_{U_i}(u_{i,N+1/2}) = 0, and\ F_{U_i}(u_{i,1-1/2}) = 1$
- The expected value of any stochastic function (cost function in this case) C is therefore:

$$E[C] = \sum_{i=1}^{M} C_i Pr(\bar{u}_i)$$

Figure 6 shows the sampling points (1764 points) resulting from a grid sampling approach over three monovariate parameters sampled respectively from uniform (Variable_U), lognormal (Variable_LN), and normal (Variable_N) distribution. Figure 7 shows instead the sampled value of a goal function set equal to the product of the lognormal and normal distributions. The sampling strategy used was equal probability for the monovariate described by the normal and lognormal distribution while equally spaced for the one characterized by the uniform. The result of this approach could be noted in Figure 7 where clearly the density of the points decrease approaching low probability regions.

The grid sampling represents a very practical tool for engineers to explore the parametric response of the system, unfortunately becomes very expensive with the growing number of dimensions posing clear computational challenges.

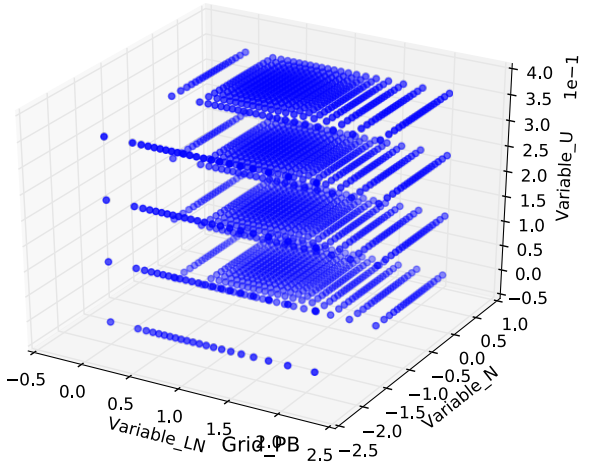

Figure 6: Scattered plot of the location of the sampling point resulting from a grid sampling over three monovariate parameters

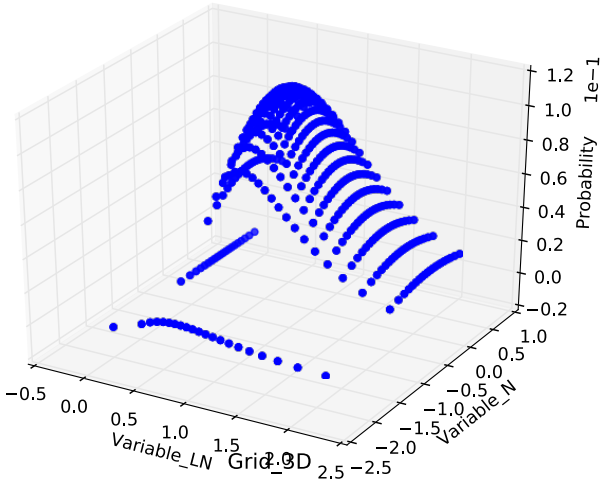

Figure 7: Probability values sampled by the Grid Sampling

2.3.4 Stratified Sampling

Stratified sampling is a class of methods that relies on the assumption that the input space can be separated in regions (strata) based on similarity of the response of the system for input set within the same strata. Following this assumption the most rewarding (in terms of computational cost vs. knowledge gained) sampling strategy would be to place one sample for each region. In this way the same information is not collected more than once and the all the prototypical behavior are sampled at least once.

In a manner very similar to the one described for the grid sampler, RAVEN allows the construction of a grid on the input parametric space either with respect to the probability associated to the random variate (CDF) or to the value itself of the variables (same limitations of the grid sampler apply). The stratification is build by assuming that there should never be more than a strata characterized by the same range of any of the random variate. This constrain allows many possible solution for the construction of the strata, one

is randomly selected and then one point is sampled from within each strata. This process is exemplified in Figure 8.

The probability associated to each point sampled is equivalent to the probability associate to the strata of which the point is representative. The computation of the risk integrals is exactly the same as in the case of the grid sampling except for the fact that the probability should account for the whole probability associated to each strata and therefore renormalized as it follows:

$$Pr(u_i) = \frac{\prod_{j=1}^{N}\left(F_{U_{i,j}}(u_{i,j+1}) - F_{U_{i,j}}(u_{i,j})\right)}{\sum_{i=1}^{points}\prod_{j=1}^{N}\left(F_{U_{i,j}}(u_{i,j+1}) - F_{U_{i,j}}(u_{i,j})\right)}$$

Where the numerator is introduced to renormalize the probability of a point so to carry along the probability of the whole strata it is representing.

The most classical implementation of stratified sampling in PRA analysis is the Latin Hypercube Sampling (LHS) where the strata are build based on equal probability regular Cartesian partition of the parametric input space. In this case each parameter is partitioned in M equal-probability range and the probability associated to each sample is $\frac{1}{M}$. Figure 9 shows the sampling points (100 points) resulting from a LHS over three monovariate parameters sampled respectively from uniform (Variable_U), lognormal (Variable_LN), and normal (Variable_N) distribution. Figure 9 shows instead the sampled value of a goal function set equal to the product of the lognormal and normal distributions. Compared to the other samplers up to now examined the LHS present the most scattered pattern and it is regarded in the PRA community as the most efficient with respect to the trade off between computational cost and information generated.

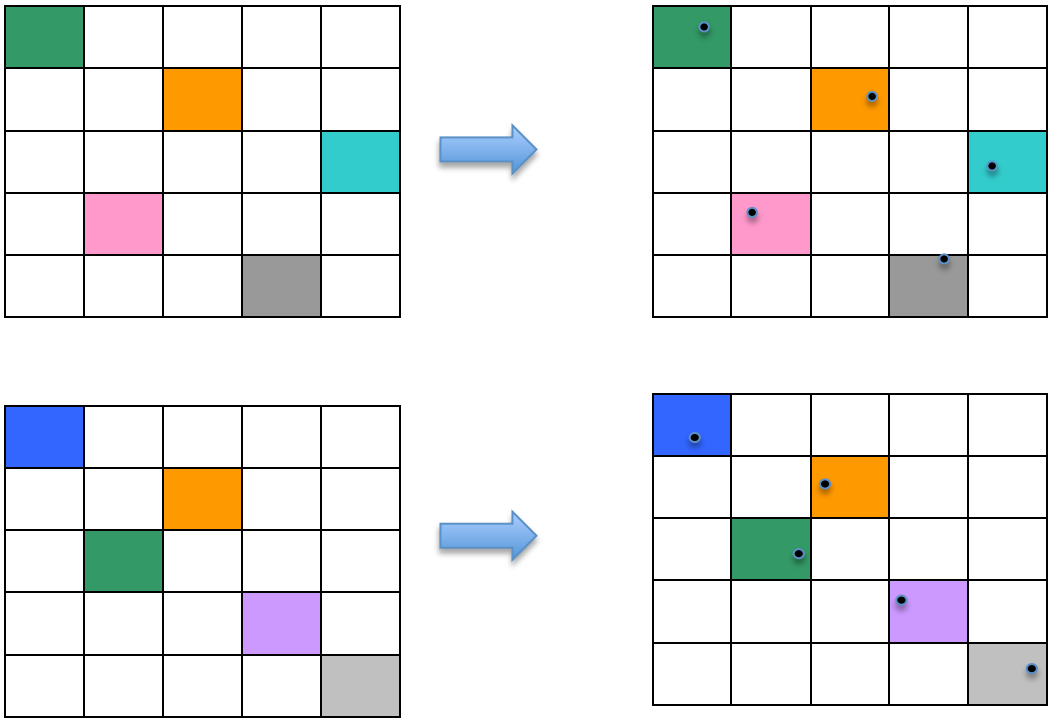

Figure 8: On the left column two possible stratifications while in the right column the corresponding possible sampling locations.

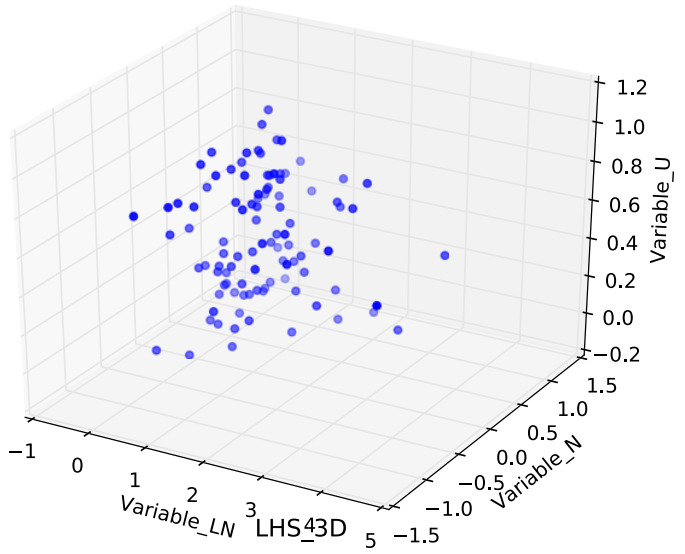

Figure 9: Scattered plot of the location of the sampling point resulting from a LHS over three monovariate parameters

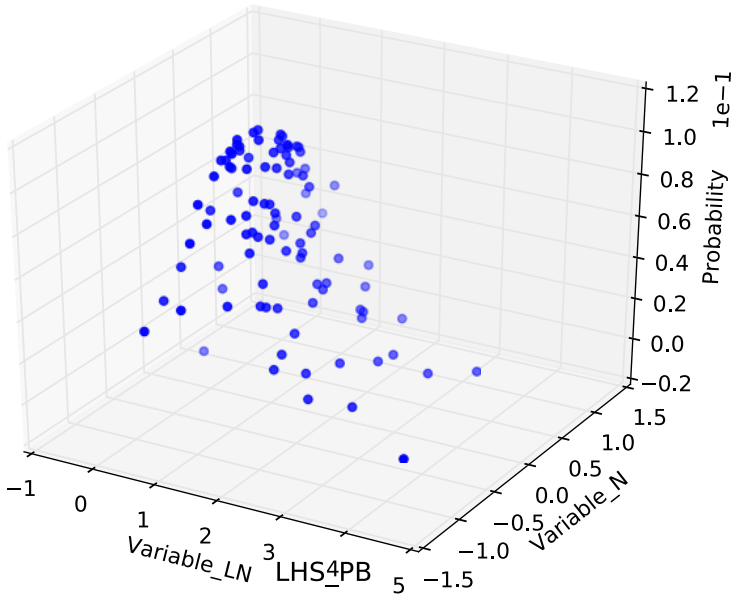

Figure 10: Probability values sampled by the Grid Sampling

2.3.5 Adaptive Sampling

2.3.5.1 The Limit Surface

One of the more advanced options that RAVEN offers is goal oriented sampling strategies for the research of limit surfaces [12]. To properly explain which type the information available by these techniques it is useful to start from the characterization of limit surfaces. Without entering in a mathematical description of the concept of limit surface it is possible to describe the limit surface as a hyper-surface, in the realization space of the stochastic parameters of the system, along which a specific goal function assumes an imposed value.

If, as in the example in Section 2, the goal function is:

$$C(\bar{x}) = \begin{cases} = \delta(t_{end}) \; if \; reactor \; is \; damaged \\ = 0 \; if \; recator \; is \; not \; damaged \end{cases},$$

the image of the limit surface could be conveniently defined as the surface, in the realization space, where $|\overline{\boldsymbol{\nabla}}\boldsymbol{C}(\overline{\boldsymbol{x}})| = \infty$, which simply is hyper surface that separates failure regions to success regions. The information content of the limit surface image is rather minimal because it just illustrates a property of the goal function with respect the status of the system but it could be conveniently noticed that the integration domain of the risk integral is the volume within the limit surface image ∂V_L:

$$R = \int_{S \cap reactor \; damaged} f_{\bar{X}}(\bar{x}, t_{end}) d\bar{x} = \int_{V_L} f_{\bar{X}}(\bar{x}, t_{end}) d\bar{x},$$

where the image of the limit surface ∂V_L satisfies $|\overline{\nabla} C(\partial V_L)| = \infty$.

Now the discussion will be narrowed to system where the probabilistic behavior could be studied only as a function of uncertainty in the model parameters and initial condition. The limitation of this assumption are described in [13] but generally are fairly acceptable for monovariate variable while might get more complex for multivariate or parametric distributions and the analysis of such cases it is still under investigation.

Under the above-described assumption, the phase space coordinate of the system at any moment in time is a function of $(\overline{\boldsymbol{x}}_{\boldsymbol{0}})$, which represents the realization of the initial conditions and stochastic parameters characterizing the system. Therefore the system behavior could be now described deterministically by:

$$\bar{x}(t) = h(\bar{x}_0, t),$$

where $h(\bar{x}_0)$ is the mathematical model representing the system, once the initial condition and the uncertain parameters are chosen. The probability propagate according:

$$f_{\bar{X}}(\bar{x}) d\bar{x} = f_{\bar{X}_0}(\bar{x}_0) d\bar{x}_0,$$

which reads that the probability of the system being in a particular state $\overline{\boldsymbol{x}}$ is equal to the probability of the initial condition and parameters to be such that leads to the prescribed state (this is a simplification that assumes h being injective, a more generic description could be found in [14]).

As a consequence the risk integral could be evaluated as:

$$R = \int_{h^{-1}(V_L)} f_{\bar{X}_0}(\bar{x}_0) d\bar{x}_0,$$

where $h^{-1}(V_L)$ is the pre-image of V_L and therefore $h^{-1}(\partial V_L)$ is the limit surface.

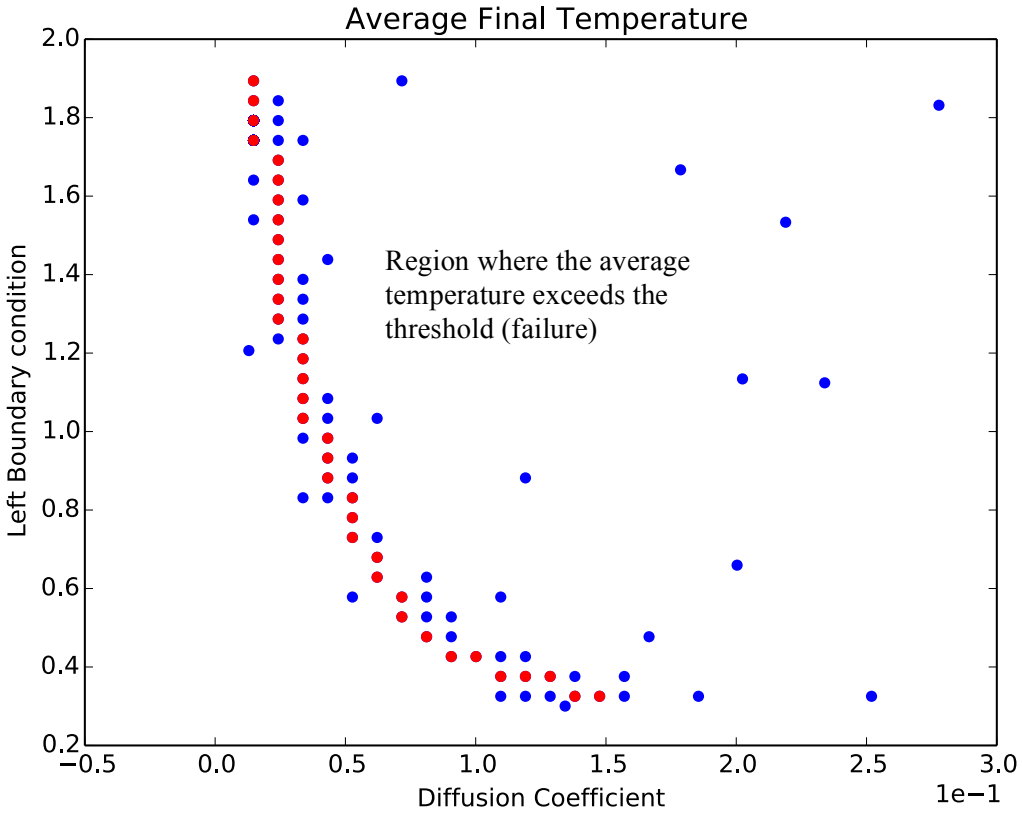

Figure 11: Limit surface example (red dots)

Figure 11 shows an example of limit surface. The example refers to the solution of the time dependent heat conduction equation in one dimensional slab:

$$\begin{cases} \dfrac{\partial T}{\partial t} = D\dfrac{\partial T}{\partial x} \\ T(x = x_L, t = 0) = T_{0,L}{}' \\ T(x = x_R, t = 0) = T_{0,R} \end{cases}$$

where $T_{0,L}$ and $T_{0,R}$ are respectively the left and right boundary conditions (on temperature), D the diffusion coefficient and T the temperature. The goal function is equal 1 if the average temperature in the slab after 20 sec exceeds the threshold value, 0 othervise. The limit surface separates the input space (diffusion coefficient and left boundary condition) depending on the value of the goal function.

In conclusion a limit surface is a hyper-surface discriminating the input space coordinates (initial condition and model parameters) depending on the evolution that the system will have with respect the value of a given cost function.

The knowledge of the limit surface allows a fast evaluation of risk functions, provides information concerning which uncertainty is mostly relevant to risk increase/decrease, defines safe areas to be explored for parametric operational optimization and risk reduction. Unfortunately the search of a limit surface in terms of computational effort is very expensive.

A brute force approach would be to build an N-dimensional grid on the input space and sample each point. The number of point in the grid would be proportional to the degree of accuracy sought and would hit rather fast a prohibitive number. To avoid such a situation RAVEN uses acceleration schemes based on surrogate models that are used to predict the location of the limit surface so to guide the exploration of the input space avoiding regions far from the frontier sought.

2.3.5.2 Surrogate Models

In the literature, there are several definitions for surrogate models and/or reduced order models and/or supervised learning process and they often overlaps. For the purpose of this technical report, a surrogate model is a mathematical model that is trained to predict the response 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 surrogate model. More specifically, in our case, the surrogate model is trained to emulate a numerical representation of the physical system that we assume possess a high degree of fidelity but it is also very computational expensive to realize. Two general characteristics of surrogate models will be assumed true in the remaining of this discussion even exceptions are possible:

1. The higher is the number of realizations in the training sets the higher is the accuracy of the prediction of the surrogate model. This is assumed true although some of the surrogate models used might be subject to the over-fitting issues. Because this a phenomena that is highly dependent on the surrogate model type, and RAVEN posses a large number of options available, the reader should consult specific literature on this subject depending on the problem to be solved.
2. The smaller is the size of the input domain with respect the variability of the system response projected on the cost function, or vice versa, the smoother is the response of the system projected on the cost function within the input domain, the more likely the surrogate model will be able to represent the risk function.

Given the fact that most of the time the cost function assume the form of a characteristic function of a certain domain (e.g. failure/success) in the phase space, in the development of the RAVEN code it has been given priority to the introduction of a class of supervised learning algorithms that are usually referred to as classifier. In essence classifiers are surrogate models specialized to represent a binary response of the system (failure/success).

The first class of classifier introduced has been the Support Vector Machines [15] with several different kernels (polynomial of arbitrary integer order, radial basis function kernel, and sigmoid) followed by a nearest-neighbor based classification using a K-D three search algorithm [16]. All those supervised learning algorithms have been imported via an Application Programming Interfaces (APIs) with the scikit-learn [17] library. It is foreseen to import soon the whole library of supervised learning methods from scikit-learn, as also the N-Dimensional spline and the inverse weight methods, which are currently available for the interpolation of N-Dimensional PDF/CDF.

2.3.5.3 The Searching Algorithm

The limit surface searching algorithm is rather straightforward and could be described by the following steps:

1. A limited number of point in the input space $\{\bar{x}_0\}_i$ are selected via one of the previously described sampling strategies (stratified, grid and Monte Carlo sampling)
2. The RELAP-7 code is used to compute the status of the system for the set of point in the input set: $\{\bar{x}(t)\}_i = h(\{\bar{x}_0\}_i, t)$
3. The Goal function (Boolean function) is evaluated at the phase space coordinate of the system: $\{g\}_i = G(\{\bar{x}(t)\}_i)$
4. The set of pairs $(\{\bar{x}_0\}_i, \{g\}_i)$ are used to train a surrogate model
5. The surrogate model (SM) is used to predict the value of the goal function on a regular Cartesian grid in the domain space (the mesh size depend on the convergence requested by the user):

$$SM(\{\bar{x}_0\}_j) \approx \{c\}_j, j = 1, \dots, M \text{ points on the Cartesian grid}$$

6. The values of the goal function are used to determine the limit surface location based on the change of values of $\{c\}_j$:

$$\{c\}_j \rightarrow \partial V_L$$

7. The position of the limit surface is compared with the one at the previous iteration (This step is skipped at the first iteration), if no changes are detected the iterations stop otherwise a new point need to be identified in the input space
8. The point located on the limit surface that is the farther from all the other already selected point in the input space is added to the $\{\bar{x}_0\}_i$ set and the process restart from point 2

The iteration scheme is graphically presented in Figure 12.

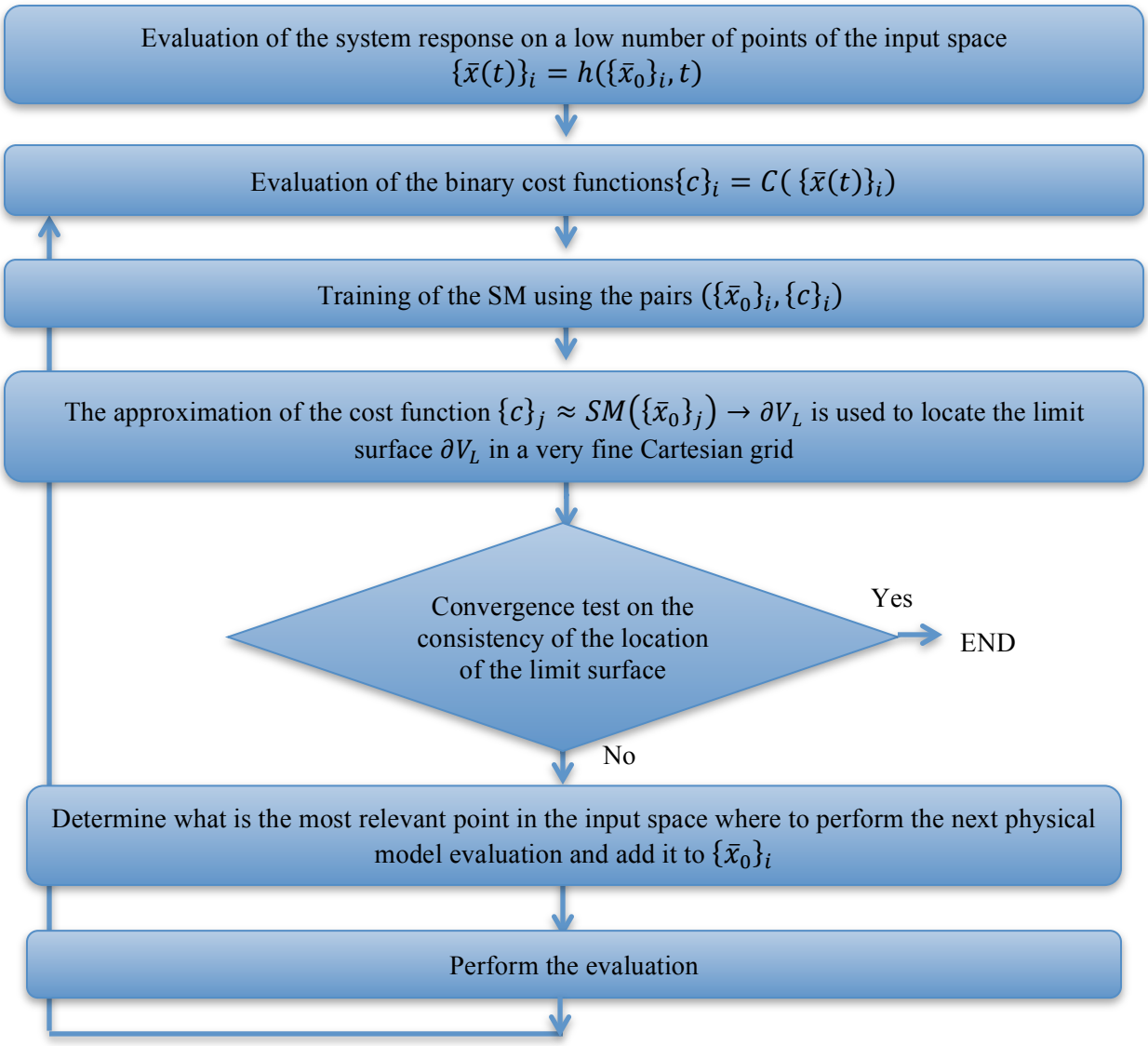

Figure 12: Limit Surface Searching Scheme

2.3.5.4 Computation of the Probability Associated

Once that the limit surface have been found the information concerning its location have been fully captured by the SM that is fully trained to locate the surface. The computation of the risk integrals is therefore performed used the M points on the Cartesian grid as it would be for a grid based sampler (see 2.3.3) where the evaluation of the cost function is replaced by its approximation by the SM: $C(\{\bar{x}(t)\}_i) \approx SM(\{\bar{x}_0\}_i)$ leading to the following expression:

$$E[C] = \sum_{i=1}^{M} SM(\{\bar{x}_0\}_i) Pr(\bar{u}_i)$$

3. BWR SBO DEMO

A simplified BWR plant system model has been built based on the parameters specified in the Organization for Economic Cooperation and Development (OECD) turbine trip benchmark problem [18]. The reference design for the OECD BWR Turbine Trip benchmark problem is derived from Peach Botom-2, which is a General Electric-designed BWR-4 NPP, with a nominal thermal power of 3,293 MW.

Figure 13 and Figure 14 show the schematics of the simplified BWR plant system that has been modeled through RELAP-7/RAVEN. The reactor vessel model consists of the down comer model, the lower plenum model, the reactor core model, the upper plenum model, the separator dryer model, the steam dome model, the main steam line model, the feed-water line model, the primary pump model, the RCIC turbine model, the RCIC pump model, and the wet well model.

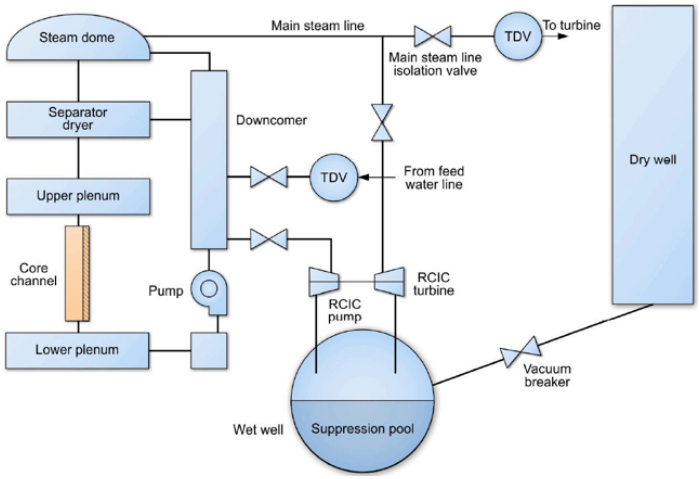

Figure 13: BWR scheme

A core channel model (i.e., flow channel with heat structure attached to it) was used to describe the reactor core. Each core channel represents thousands of real cooling channels and fuel rods. To speed up the transient simulation, only one core channel was used to represent the entire core; bypass flow was ignored. The lower plenum, upper plenum and steam dome are modeled with branch models. External to the reactor vessel, the main steam line is connected to the steam dome. A time dependent volume is attached to the main steam line to provide the necessary boundary conditions for the steam flow. A feed-water line is connected to the down comer model. A time dependent volume is attached to the feed-water line to provide the necessary boundary conditions for the feed-water. The safety injection system includes the RCIC turbine and pump, as well as the containment wet well and dry well. Valves are placed at various locations to provide the flow control functions of the plant system.

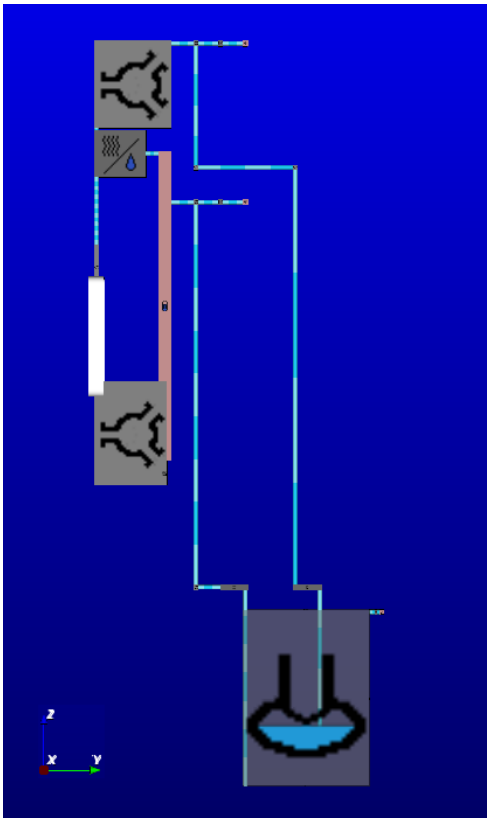

Figure 14: BWR plant visualization in RAVEN

Notably missing from this simplified BWR model are the jet pumps and the recirculation loops that allow the operator to vary coolant flow through the core and change reactor power. Instead, for this case study, a pump model is used to represent the functions of the jet pumps and the recirculation loops.

The following paragraph provides more detailed information on the plant geometry and parameters used to derive this demo.

The Peach Bottom-2 reactor core consists of 764 fuel assemblies. The initial cycle was selected as the reference design cycle for the simulations done in this report. In the initial cycle, 7×7 fuel rod lattice type assemblies with no water rods were loaded. The active core height specified was 3.6576 m. For ease of preparing the input file, we used 3.66 m as the active core height in our calculations. The fuel assembly and fuel rod geometry data were taken from reference [18] and shown in Table 2.

Table 2: Core model parameter and fuel rod geometry data.

Core thermal power (MW)	3,293
Core height (m)	3.66
Core flow area (m^2)	7.8
Fuel pellet diameter (cm)	1.21158
Gap thickness (cm)	0.01524
Clad outer diameter (cm)	1.43002
Fuel rod pitch (cm)	1.8745
Number of fuel rods per assembly	49
Assembly pitch (cm)	15.24
Heat transfer surface area per unit fluid volume	235.4927
Hydraulic diameter (cm)	1.3597

The major parameters required to build the simplified BWR plant system configurations also were obtained from reference [18] and some key data are shown in Table 3.

Table 3: Major component parameters for the simplified BWR plant configuration

Component Name	Volume (m^3)	Area (m^2)	Axial Elevation (Top) Relative to the Bottom of the Vessel (m)
Lower Plenum	61.48	11.64	5.28
Reactor Core	28.55	7.8	8.94
Upper Plenum	26.99	14.36	10.82
Separator Standpipe	10.69	3.93	13.54
Separator Dryer (S/D)	19.30	10.27	15.42
S/D Steam Outlet Pipe	0.393	3.93	15.42
S/D Liquid Discharge Pipe	3.93	3.93	14.48
Steam Dome	178.19	26.19	22.32
Main Steam Line	2.64	1.32	18.92
Down Comer	201.30	15.00	15.52
Feedwater Line	3.96	1.32	12.52
Wet Well Water Space	3570.00	892.50	-12.00 (bottom)
Wet Well Gas Space	3570.00	892.50	-4.00 (top)
RCIC Turbine	-	-	-3.00
RCIC Pump	-	-	-3.00

3.1 Demo Description

The scenario considered is a grid related loss of off-site power (LOOP) event immediately followed by the loss of the emergency diesel generators (DGs). Such event is known as station black-out (SBO) initiating event. Due to the complete loss of AC power, after the reactor operators successfully scram the reactor, the cooling of the core and the removal of the decay heat is performed by using high-pressure cooling system, the RCIC system, which removes the steam from reactor pressure vessel (RPV) and dump it into the suppression pool (i.e., wet well). At the same time, RCIC injects cool water from the suppression pool back to the RPV. This procedure is followed until AC power is restored either by recover off-site power or by fixing issues related to the DGs. From a stochastic point of view, several random variables have been introduced to represent the probabilistic evolution of the system and are here reported:

- Off-site power recovery time (see Figure 15): lognormal (mean=2.66 and sigma=2.0) truncated within the interval [20.0, 600.0] s
- DGs recovery time (see Figure 16): Weibull (k=0.745 and lambda=120) truncated within interval [1.0, 600.0] s
- Burn-up of the fuel: exponentially distributed with lambda 0.05 within the interval [0.0, 60.0] GWd/MtHM
- Failure temperature of the cladding: 1-dimensional triangular parametric distribution that describes the PDF of the temperature at which the clad fails as a function of Burn-up (see Figure 17). Such distribution models the fact that clad failure likelihood increases for higher Burn-Up values. A more detailed description of the mathematical model is provided in the following paragraph.

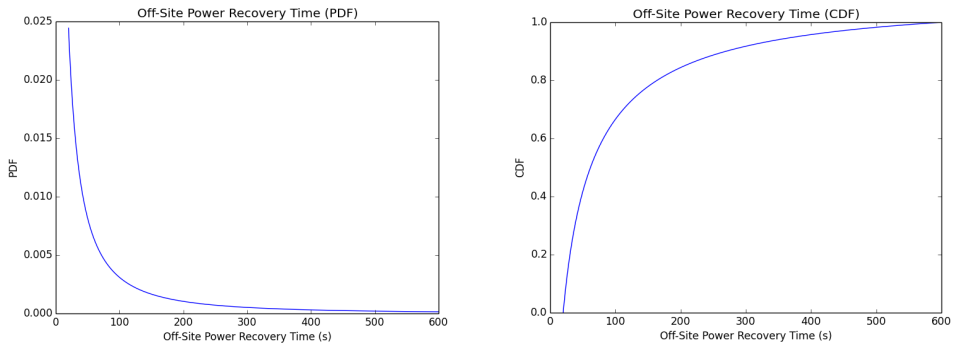

Figure 15: PDF and CDF for Off-Site Power Recovery Time

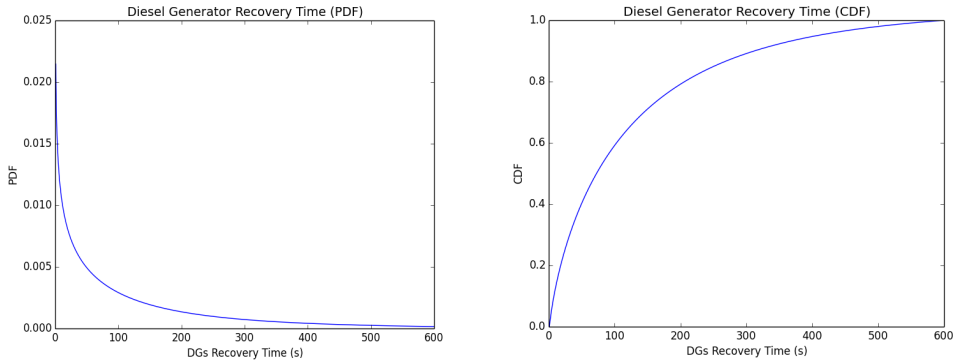

Figure 16: PDF and CDF for Diesel Generator Time

As previously mentioned, handling multivariate and/or parametric distributions can be challenging from a sampling point of view. The usage of those distributions, in whatever sampling strategy the user decides to employ, needs to be carefully handled. The main problem is related to the fact that the inverse of a multivariate and/or parametric CDF is not a bijection (it is represented by an hyper-surface); this does not allow the usage of a strategy based on the concept of point sampler (the system code is run for specific values of the random variables that are set externally) and requires a different approach. RAVEN, in such cases, samples the CDF between [0,1] and checks, during the simulation, if the CDF threshold gets crossed in consequence of the evolution of the system.

Here, it is shortly described how the 1-dimensional parametric PDF, considered in this demo, has been handled:

- Random variable: clad failure temperature with triangular distribution ($T_{C,F}$).
- Parameter: burn-up level (Bu).
- Probability distribution function:

$$T_{C,F} \sim pdf\left(T_{C,F}\right) = \begin{cases} 0, & T < T_{C,F,min} \\ \dfrac{2\left(T_{C,F} - T_{C,F,min}\right)}{\left(T_{C,F,max} - T_{C,F,min}\right)\left(T_{C,F,peak} - T_{C,F,min}\right)}, & T_{C,F,min} < T < T_{C,F,peak} \\ \dfrac{2\left(T_{C,F,peak} - T_{C,F}\right)}{\left(T_{C,F,max} - T_{C,F,min}\right)\left(T_{C,F,max} - T_{C,F,peak}\right)}, & T_{C,F,Peak} < T < T_{C,F,max} \\ 0, & T_{C,F,Peak} > T \end{cases},$$

with the parametric dependence from the Burn-up accounted by:

$$T_{C,F,max} = 1699.81e^{-0.092354\mathrm{Bu}},$$
$$T_{C,F,min} = 1255.37e^{-0.092354\mathrm{Bu}},$$
$$T_{C,F,peak} = 1477.59e^{-0.092354\mathrm{Bu}}.$$

Where $T_{C,F,min}$, $T_{C,F,peak}$, $T_{C,F,max}$ are respectively the minimum, the most probable and the maximum values of the clad failure temperature.

- Before each simulation starts, a random number between [0, 1] is generated as threshold of the clad failure temperature CDF (cdf_{Th}).
- Each time RELAP-7 advances the solution in time, the Burn-up level and the clad temperature, coming from the simulation, are used to perform a test on the CDF value:

$$cdf\left(T_{C,F}, Bu\right) \lessgtr cdf_{Th}$$

- Depending on exceeding/not exceeding the CDF threshold the simulation evolution is modified according:

$cdf\left(T_{C,F}, Bu\right) < cdf_{Th}$: the simulation goes on without alteration,

$cdf\left(T_{C,F}, Bu\right) \geq cdf_{Th}$: the simulation stops as consequence of the clad failure detection.

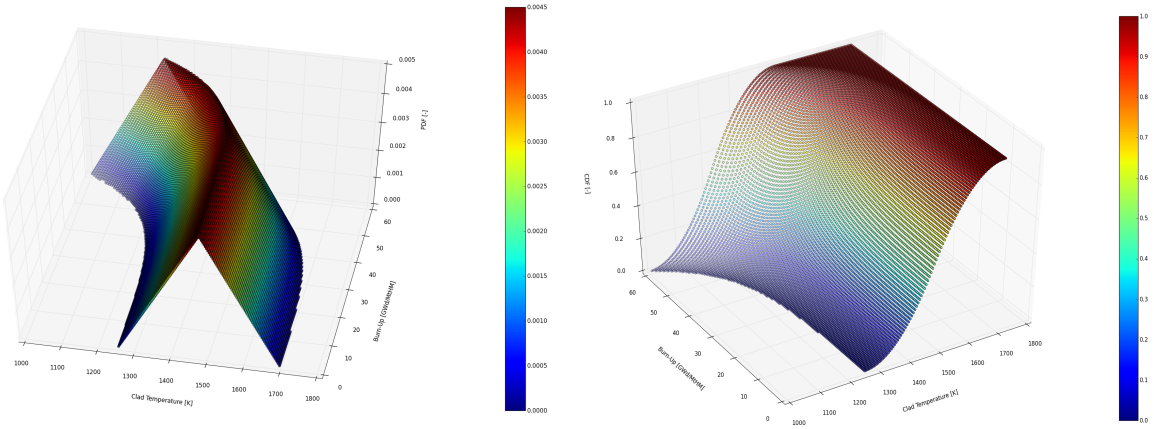

Figure 17: PDF and CDF for clad failure as function of Burn-Up and Temperature[a]

3.1.1 Analysis Performed

The main goal of this demo is to show some of the capabilities developed in the last year by the RAVEN team. As already mentioned, the intense development of RAVEN resulted in the addition of all the state-of-art sampling strategies along with all the capabilities to perform probability and uncertainty quantifications.

The SBO scenario has been analyzed through the 4 principal sampling strategies (Dynamic Event Tree is subject of a future milestone report): Monte Carlo, Grid sampling, Latin Hypercube and Adaptive sampling. Since the final scope was to perform a comparison among these methodologies, it has been decided to set the maximum number of samples, for each sampler, to a common upper bound of 1224.

It is important to note that, since the RELAP-7 computational performances are still in a developing stage, the RAVEN team artificially shorted the simulated time (maximum Simulation Time = 450

[a] Plots generated by RAVEN code

seconds) by acting on unconventional parameters, such as friction factors and decay heat curve. These actions do not affect the validity of the goal of this report that, as already mentioned, is focused on RAVEN algorithm capabilities.

3.2 Results

3.2.1 Monte Carlo

The Monte Carlo sampling has been performed setting a limit of 1224 calculations. As it will be pointed out later, the Monte Carlo is the only strategy that involved a sampling of the distributions (CDFs) associated to the uncertain parameters mainly using the transformation method described in 2.3.1.1. In fact, the other sampling strategies use a structured discretization of the input space either in value or cumulative probability and probability distribution functions are used to compute the failure probability as described in 2.3.3, 2.3.4 and 2.3.5.

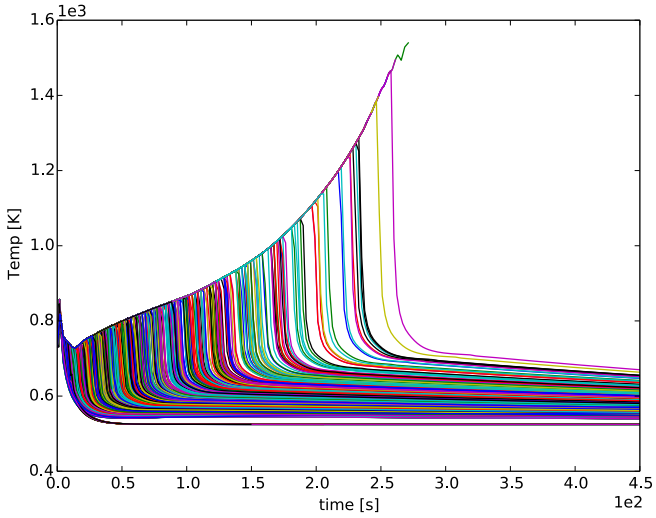

Figure 18: Clad Temperature Evolution Monte Carlo

Figure 18 shows the clad temperature evolution for all the histories simulated. Right after the reactor scram, the temperature of the clad starts decreasing until the primary pumps, in coast-down condition, are completely stopped. When they stop, the temperature starts rising until the auxiliary cooling system is possibly restored, leading to a further cooling of the core and placing the NPP in safety conditions.

Figure 19 shows the histograms of the sampled Burn-up level and the threshold CDF (cdf_{Th}) for the clad failure temperature following the Monte Carlo sampling of the respectively exponential and uniform distributions. Figure 21 shows, instead, the histogram resulting from the sampling of the truncated Weibull and Log Normal distributions being respectively the PDF of the DGs and off-site power recovery time.

Figure 22 shows the distribution of the maximum temperature reached by the clad for the Monte Carlo analysis. Most of the sampled scenarios reached maximum temperatures below 1000 K, i.e., the minimum temperature that could cause the failure of the system, at high Burn-ups (see Figure 18) indicating therefore that most of the scenarios ended without failure of the clad.

Figure 21 shows the limit surface that has been generated based on the Monte Carlo sampling. Since the uncertain parameters represent a 4-D input space, three of them are explicitly treated in as 3-D coordinate and the remaining one, the Probability Threshold, is used as weight for the colors of circles plotting the limit surface. As expected, the limit surface presents itself as the intersection of two planes almost parallel, one to the DG recovery – Burn-up plane and the other to the off site recovery time –

Burn-up plane. The reason for this shape is that the recoveries of the diesel generators or the off-site power are equivalent with respect the determination of the clad temperature. As a consequence the limit surface should, in some extend, being similar to the surface representing the following mathematical expression:

$$min\{DG\ recovery\ time, Off\ site\ recovery\ time\} > T \rightarrow failure,$$
$$min\{DG\ recovery\ time, Off\ site\ recovery\ time\} < T \rightarrow success,$$

where T is a generic recovery time such as the clad temperature does not exceed the failure value. T is, obviously, a function of the cdf_{Th} and Bu. These dependencies generate the bending of the planes so that the success area (region characterized by the low values of DGs and off-site power recovery time) decreases with the increase of the Burn-up and increase with the increase of the CDF threshold chosen for the clad failure temperature (cdf_{Th}).

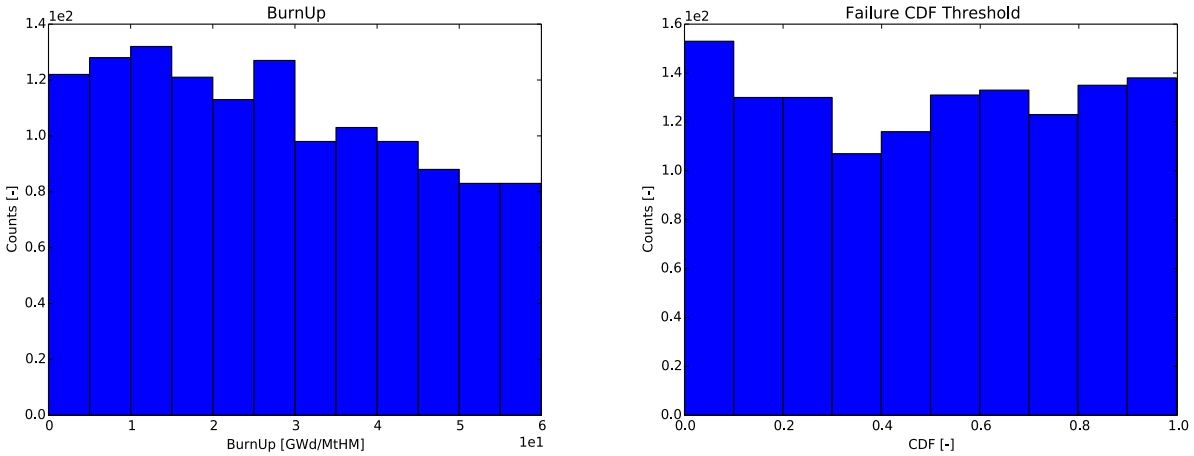

Figure 19: Burn-Up and Probability threshold histograms

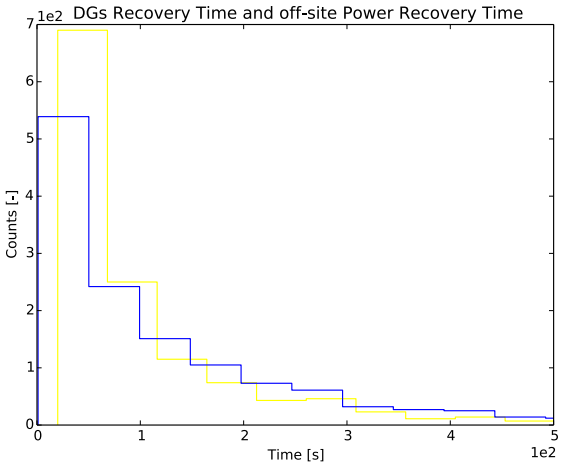

Figure 20: Diesel Recovery Time (blue), off-site Power Recovery Time (yellow)

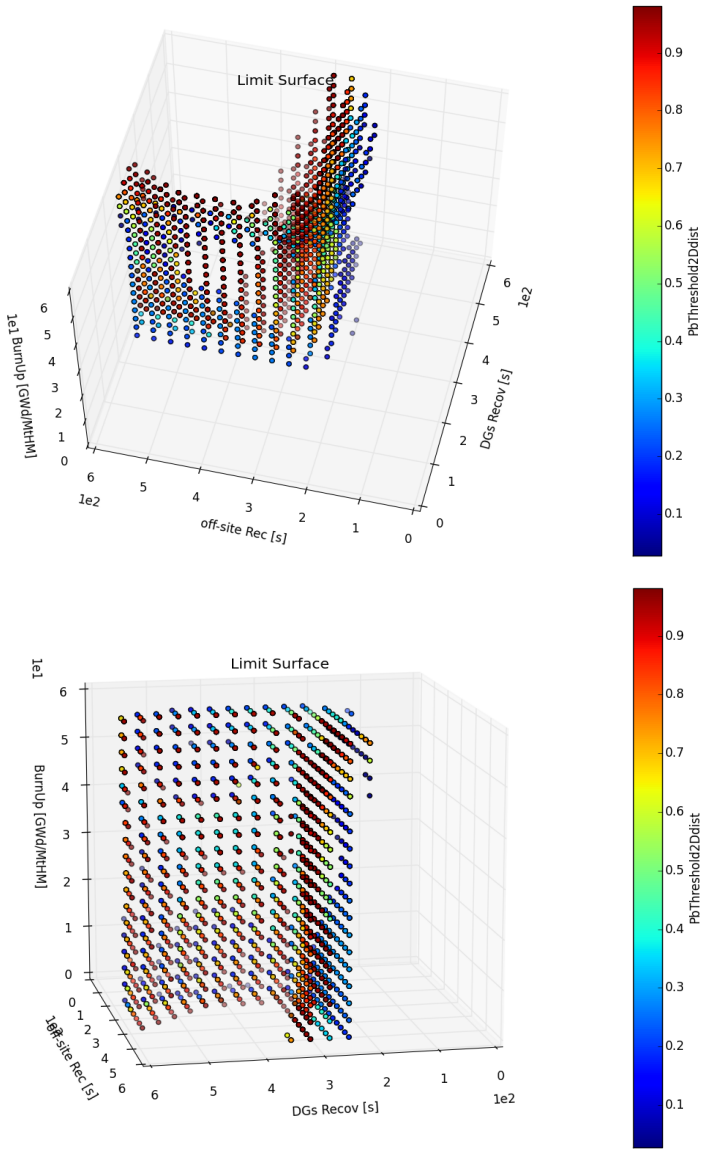

Figure 21: Views of the limit surface obtained from the Monte-Carlo sampling strategy

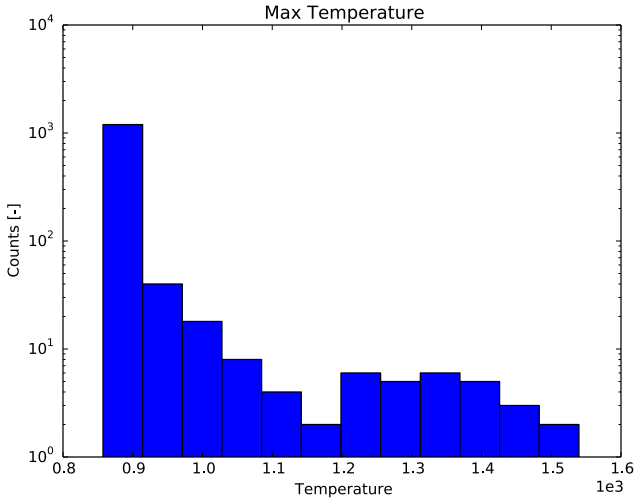

Figure 22: Max clad temperature histogram

In the following tables the probability of failure (Table 4) and the correlation matrix (Table 5) are reported. The correlation matrix shown in Table 5 highlights, as expected, the importance of the AC power recovery time (either by DG repair or off-site power recovery) followed by Pb_threshold associated to the clad failure temperature and its burn-up value. This could be inferred by observing the correlation between the 'clad failed' and the other entries of the matrix. The correlation between the sampled variables (DG recovery time, off site power recovery time, Burn-up and CDF threshold) is negligible given the fact that they represent independent parameters.

Table 4 Failure Probability Monte Carlo

Failure Probability	**1.31E-02**
Sigma	3.23E-03

Table 5 Correlation Matrix Monte Carlo

Correlation Matrix	DGs Recovery Time	Off-Site Power Recovery Time	Burn-Up	Pb Threshold	Clad Failed
DGs Recovery Time	1.00E+00	-3.74E-02	-4.56E-03	-2.41E-02	2.35E-01
Off-Site Power Recovery Time	-3.74E-02	1.00E+00	9.59E-03	-3.65E-02	2.70E-01
Burn-Up	-4.56E-03	9.59E-03	1.00E+00	3.05E-02	2.35E-02
Pb Thresholds	-2.41E-02	-3.65E-02	3.05E-02	1.00E+00	-9.03E-02
Clad Failed	2.35E-01	2.70E-01	2.35E-02	-9.03E-02	1.00E+00

3.2.2 Grid Sampling

In the first part of this report it has been highlighted that sampling performed on a Cartesian grid is the simplest algorithm can be used for exploring the input space characterized by a set of uncertain parameters. In this section the SBO scenario is analyzed by mean of an equally spaced value grid and the results are compared to the ones obtained by the Monte Carlo sampling strategy. The following table summarizes the discretization used.

Uncertain Parameter	# of equally spaced mesh intervals	Mesh size
Diesel Generators' Recovery Time	5	100 s
Off-site Power Recovery Time	5	100 s
Burn-Up	5	12 GWd/MtHM
2D Probability Thresholds	5	0.2

Table 6 Grid Meshing

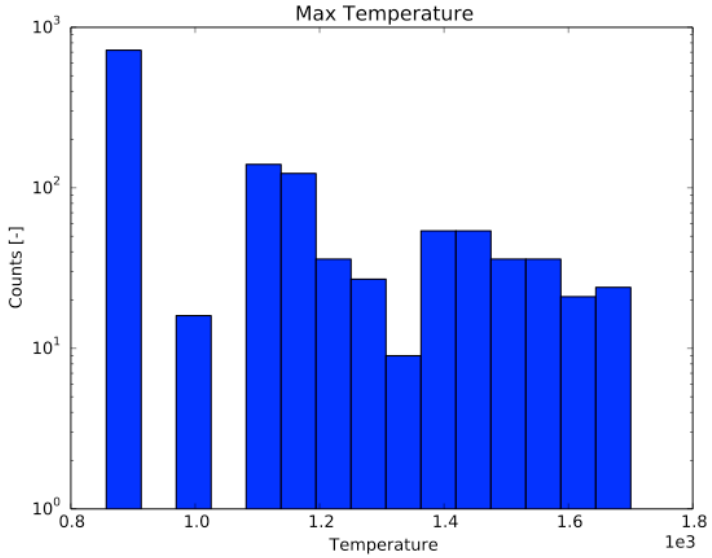

Figure 23: Max Temperature Histogram Grid

Since the grid used is equally spaced in value, the histograms of the sampled variables are not going to be reported here, not being correlated to the associated distributions. The distributions, as already mentioned, have been used for the computation of the failure probability only.

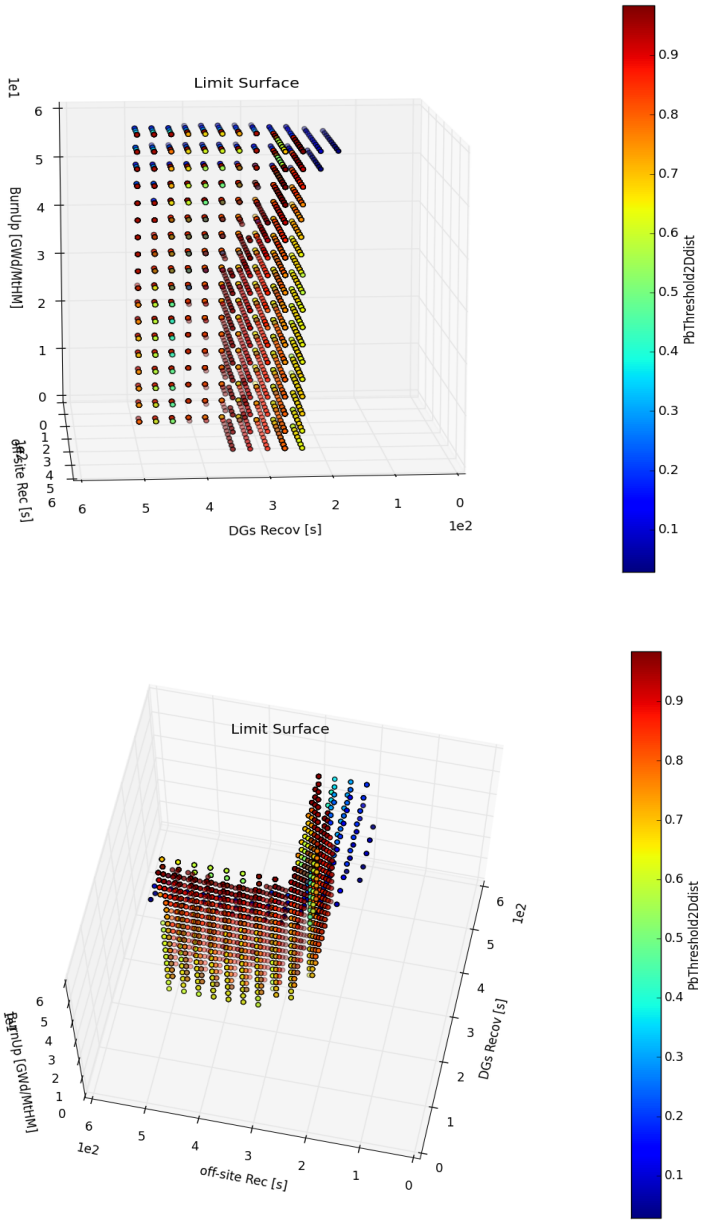

Figure 24: View of the limit surface obtained from the grid sampling strategy

Figure 23 shows the distribution of the maximum temperature reached by the clad for the Grid based analysis. As it can been seen, most of the sampled scenarios reached maximum temperatures below of 1000 K, i.e., the minimum temperature that could cause the failure of the system, at high Burn-ups (see Figure 18) meaning that also in this case, as for the Monte Carlo, the sampling was focused on areas of the input space mostly leading to successful scenarios. From this figure it can be also noted that the grid, as expected, given the low number of sampling points, is not very accurate in the exploration of the input space. In fact it leads to a low coverage of some of the bins. On the positive side this approach enlarges the range of covered temperatures with respect the Monte Carlo approach (Figure 22) providing a better explorative approach.

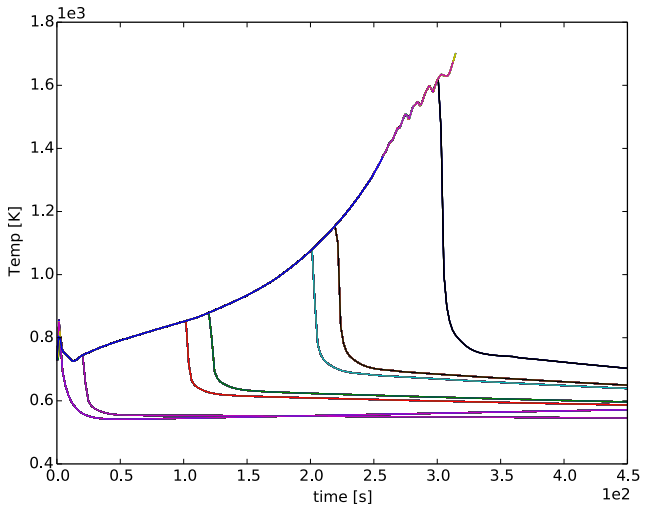

Figure 25: Clad Temperature temporal profile for the Grid sampling

Figure 24 shows the limit surface generated using the Grid sampling strategy. As already mentioned, since the uncertain parameters represent a 4-D input space, three of them are explicitly plotted in the 3-D space while the forth one, the Probability Threshold, affects the colors of the points lying on the limit surface. Behavior of the limit surface appears very similar to the one detected by the Monte Carlo approach, confirming the overall congruence of the two approaches. The grid approach offers a better resolution of the general trends characterizing the limit surface being more regular in its explorative strategy.

Figure 26 shows the clad temperature evolution obtained by the grid sampling. The analysis of the figure confirms the coarse exploration of the input and, consequentially, output space (as already observed in the histogram analysis reported in Figure 23**)**.

In the following tables the probability of failure and the correlation matrix, among uncertain parameters and the target parameter (Clad Failed) are reported.

Table 7 Failure probability Grid Sampling

Failure Probability	1.33E-02
Sigma	1.15E-01

Table 8 Correlation Matrix Grid Sampling

Correlation Matrix	DGs Recovery Time	Off-Site Power Recovery Time	Burn-Up	Pb Threshold	Clad Failed
DGs Recovery Time	1.00E+00	8.70E-17	-3.11E-18	-1.18E-17	5.09E-01
Off-Site Power Recovery Time	8.70E-17	1.00E+00	-1.31E-17	-8.93E-18	4.90E-01
Burn-Up	3.11E-18	-1.31E-17	1.00E+00	-6.27E-18	3.87E-02
Pb Thresholds	-1.20E-17	-2.16E-18	-6.27E-18	1.00E+00	-5.44E-02
Clad Failed	5.09E-01	4.90E-01	3.87E-02	-5.44E-02	1.00E+00

Table 7 reports the probability of failure computed through the Grid sampler. Its value has been computed following the approach reported in chapter 3 and closely matches the one computed by the Monte Carlo method.

Values indicated in Table 8 agree to the ones presented in Table 5 and, hence, similar conclusions can be inferred regarding the greater importance of the AC recovery timing.

3.2.3 Latin Hypercube Sampling

As already mentioned, in order to perform a comparison among the different sampling strategies a limit of 1224 simulations has been set. From a Latin Hypercube point of view, this has been translated using a grid of 1224 equally spaced probability discretization intervals.

Since the grid used for the latin hypercube sampling is equally spaced, in probability, the histograms of the sampled variables are not going to be reported here, not being correleted to the associated distributions. The distributions, as for the grid analysis, have been used for the computation of the failure probability only.

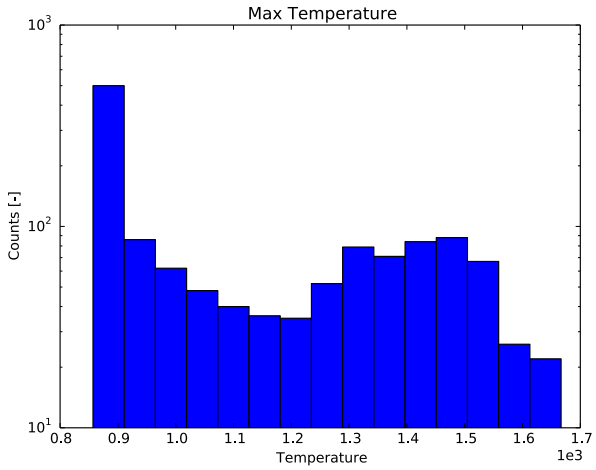

Figure 26: Max Temperature Histogram LHS

Figure 26 shows the distribution of the maximum temperature reached by the clad for the LHS analysis. Similarly to the Monte Carlo case, the outcomes are evenly distributed in whole output space. As for all the cases shown so far, most of the sampled scenarios reached maximum temperatures below of 1000 K. From this figure it can be noticed that the LHS is a good sampling strategy for the exploration of the uncertainty space.

This result is confirmed by Figure 28. That shows how the clad temperature evolution of the LHS effectively covers the possible output space.

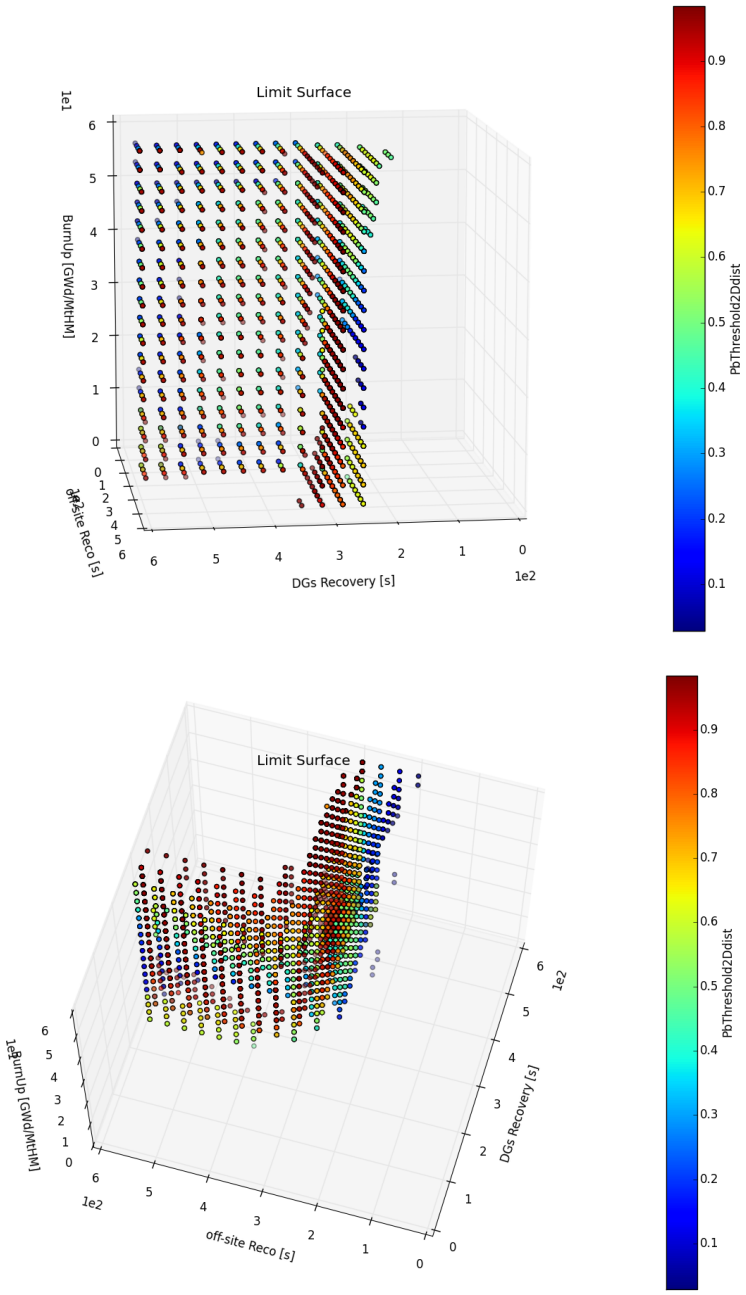

Figure 27: Views of the limit surface obtained from LHS strategy

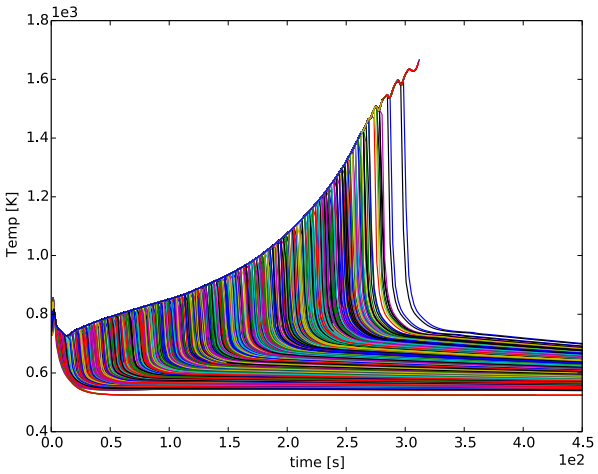

Figure 28: Clad Temperature Evolution LHS

Figure 27 shows the limit surface that has been generated based on the Latin Hyper Cube sampling. The limit surface is fairly similar to the previous reported ones confirming the results so far obtained.

In the following tables the probability of failure and the correlation matrix, among uncertain parameters and the target parameter (Clad Failed) are reported.

Table 9: Failure Probability LHS

Failure Probability	1.53E-02
Sigma	1.23E-01

Table 10: Correlation Matrix LHS

Correlation Matrix	DGs Recovery Time	Off-Site Power Recovery Time	Burn-Up	Pb Threshold	Clad Failed
DGs Recovery Time	1.00E+00	-2.78E-04	-1.71E-02	-4.40E-03	5.19E-01
Off-Site Power Recovery Time	-2.78E-04	1.00E+00	-2.23E-02	-9.93E-03	5.19E-01
Burn-Up	-1.71E-02	-2.23E-02	1.00E+00	-9.49E-03	8.69E-03
Pb Thresholds	-4.40E-03	-9.93E-03	-9.49E-03	1.00E+00	-7.10E-02
Clad Failed	5.19E-01	5.19E-01	8.69E-03	-7.10E-02	1.00E+00

Table 9 reports the probability of failure computed through the LHS sampler. Its value is in agreement with the previous reported values.

Again, as stated for the correlations matrices already shown, great importance is given to AC power recovery (either DG or off-site power) followed by the Pb_threshold associated to the clad failure temperature and burn-up values.

3.2.4 Adaptive Sampling

The adaptive sampling strategy is the most promising sampling strategy currently present in RAVEN. For this analysis, a convergence criterion based on probability has been set to a minimum tolerance of 1.0E-4. The Adaptive sampler converged in this confidence interval in ~1000 iterations (simulations).

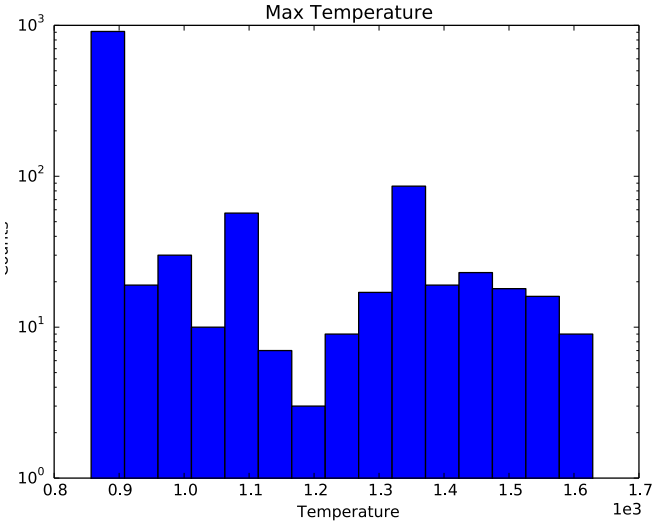

Figure 29: Max Temperature Histogram Adaptive

Figure 29 shows the distribution of the maximum temperature reached by the clad for the Adaptive sampling analysis. This figure already shows the goal-oriented behavior of the methodology; as it can be seen the right side of the histogram is more populated then the ones obtained by the other sampling approaches. This is due by the fact the adaptive strategy tends to explore the input space that most likely is close to the failure boundary, reasonably focusing around scenarios characterized by higher temperatures.

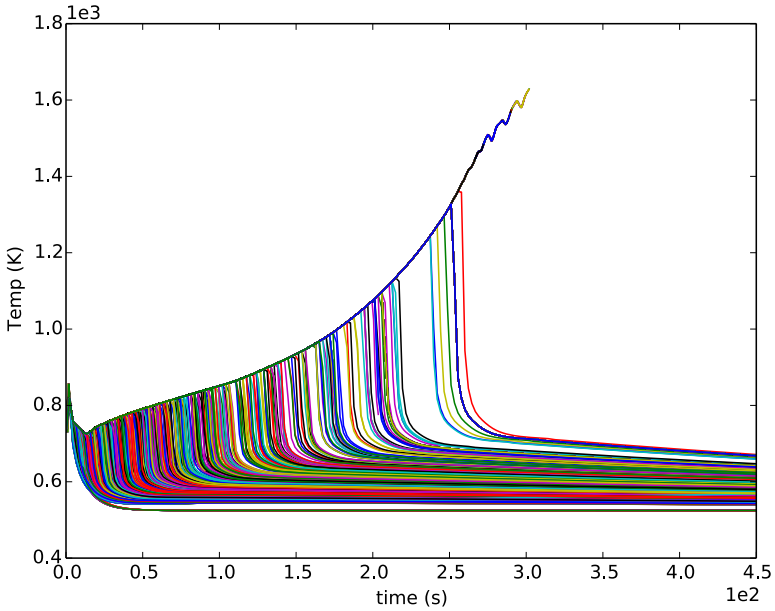

Figure 30: Clad Temperature Evolution Adaptive

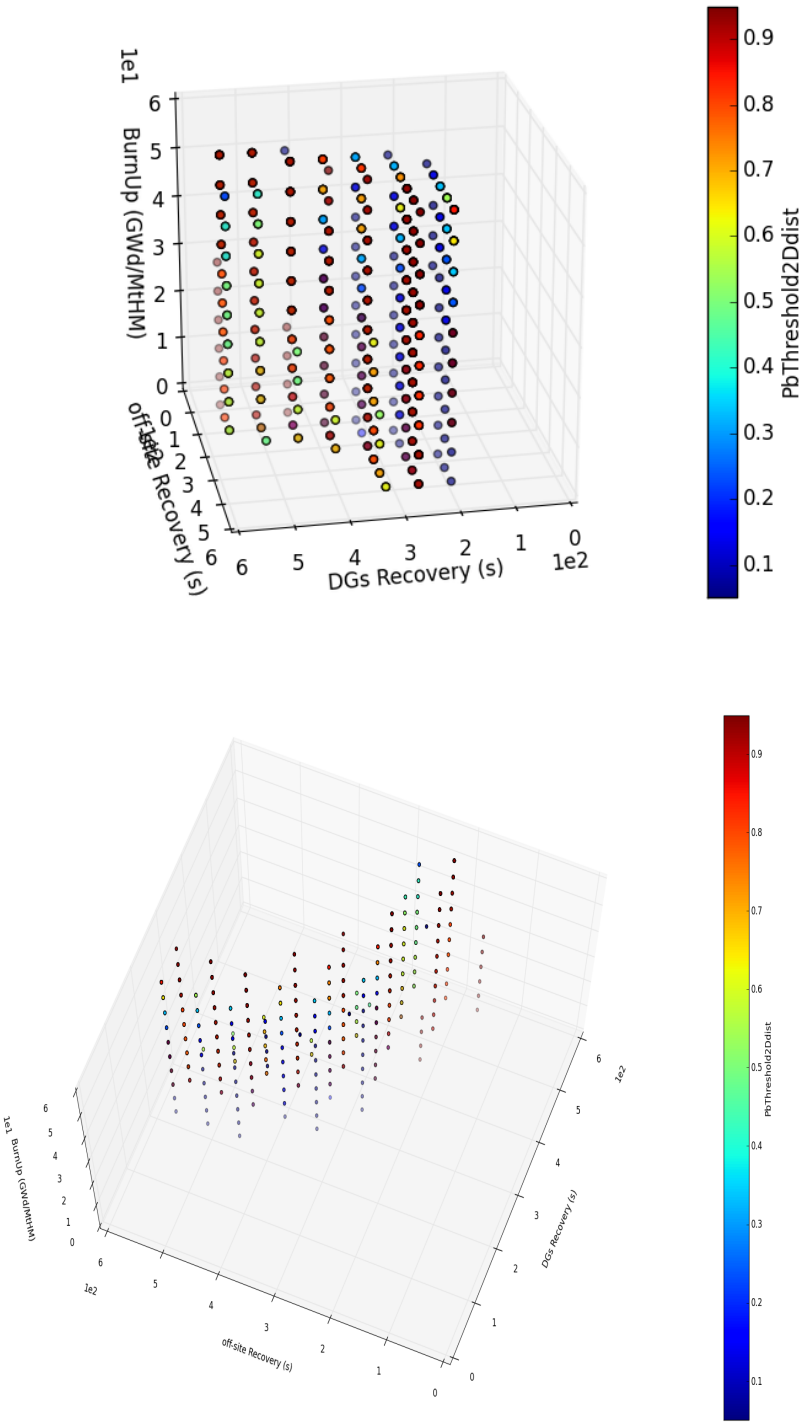

Figure 31: Limit surface e obtained from the adaptive sampling strategy

Figure 30 shows the clad temperature evolution generated by the adaptive sampling analysis. As it can be seen, there are several changes in color (indicating multiple lines overlapping) in correspondence of scenarios with high clad temperatures (right side of the plot). Once more, this is due by the intrinsic

nature of the adaptive strategy that explores the input and output space that most likely can lead to the failure of the clad.

Figure 31 shows the limit surface that has been generated using the Adaptive sampling strategy. The limit surface shows the same shape characteristics of the others already seen for the other sampling strategies. The limit surface looks clearer because the samples are highly concentrated along the limit surface itself.

The following tables show the probability of failure and the correlation matrix, among uncertain parameters and the target parameter (Clad Failed).

Table 11: Failure Probability Adaptive

Failure Probability	1.13E-02
Sigma	1.11E-01

Table 12: Correlation Matrix Adaptive

Correlation Matrix	DGs Recovery Time	Off-Site Power Recovery Time	Burn-Up	Pb Threshold	Clad Failed
DGs Recovery Time	1.00E+00	3.61E-01	2.84E-02	-4.67E-02	5.06E-01
Off-Site Power Recovery Time	3.61E-01	1.00E+00	4.85E-02	-4.49E-02	5.41E-01
Burn-Up	2.84E-02	4.85E-02	1.00E+00	-6.34E-02	2.91E-02
Pb Thresholds	-4.67E-02	-4.49E-02	-6.34E-02	1.00E+00	-4.47E-02
Clad Failed	5.06E-01	5.41E-01	2.91E-02	-4.47E-02	1.00E+00

Table 11 shows the probability of failure computed through the adaptive sampling. As it can be seen, its value is in agreement with the Monte Carlo one. The sigma associate is larger then the Monte Carlo one; this is explainable since the number of observations of failure events in the adaptive sampling is much higher.

Table 12 confirms the conclusions already stated for the other sampling strategies.

4. CONCLUSIONS

As highlighted in the first part of this report RAVEN's statistical analysis have made large progress forward. All classical and most widely used statistical methods (Monte Carlo, stratified sampling and grid-based sampling) have been implemented and successfully tested. New methodologies have been explored as the limit surface searching approach. Those new methodologies have shown great potentiality in both increase accuracy of risk estimation and reduction of computational time. More work will be needed in the future to optimize these new algorithms and provide robust error estimators. The coupling of RAVEN and RELAP-7 has been tested in conjunction with those new features of the RAVEN code for a demo of a BWR SBO core damage probabilistic analysis. During this test a sizable number of probabilistic distributions have been employed in conjunction with also a parametric distribution function that is a capability that is unique to the synergy rising from the usage of MOOSE as a common platform for the RAVEN control logic and the RELAP-7 code. All statistical analysis approach tested provides coherent results within the prescribed tolerances. The conclusion is that the theoretical bases are sounds,,

the software implementation has been proven to be solid and supporting the conclusions of the theoretical derivation. The next step will be to extend the complexity of the analysis performed and to push further the computational effort so to provide a more stringent confutation of the theoretical basis.

Overall RAVEN is proposing itself as a valid tool for a more comprehensive and also computational efficient tool to perform PRA analysis.

REFERENCES

[1] NEAMS: The Nuclear Energy Advanced Modeling and Simulation Program, Technical Report ANL/NE-13/5

[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] "Light Water Reactor Sustainability Program Integrated Program Plan, Revision 1," INL-EXT-11-23452, April 2013

[4] "RELAP-7 Level 2 Milestone Report: Demonstration of a Steady State Single Phase PWR Simulation with RELAP-7," INL/EXT-12-25924

[5] C. Rabiti, A. Alfonsi, D. Mandelli, J. Cogliati, R. Martinueau, C. Smith, "Deployment and Overview of RAVEN Capabilities for a Probabilistic Risk Assessment Demo for a PWR Station Blackout," Idaho National Laboratory report: INL/EXT-13-29510 (2013).

[6] Boost Team, http://www.boost.org

[7] A. C. Rencher, Methods of Multivariate Analysis. New York: Wiley (1995).

[8] C. Habermann, F. Kindermann, "Multidimensional Spline Interpolation: Theory and Applications", Computational Economics, Volume 30, Issue 2, pp 153-169 (2007)

[9] W. J. Gordon and J. A. Wixom, "Shepard's Method of "Metric Interpolation" to Bivariate and Multivariate Interpolation", Mathematics and Computation, vol. 32, n 141, pp 253-264, 1978.

[10] T. Itoh, K. Koyamada. Isosurface generation by using extrema graphs, In proceeding of: Visualization, 1994., Visualization '94, Proceedings., IEEE Conference on

[11] R Durrett (1995). Probability: Theory and Examples, 2nd Edition. Duxbury Press.

[12] 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

[13] C. Rabiti, D. Mandelli, A. Alfonsi, J. Cogliati, and B. Kinoshita, "Mathematical framework for the analysis of dynamic stochastic systems with the raven code," in Proceedings of International Conference of mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2013), Sun Valley (Idaho), pp. 320–332, 2013.

[14] C. Rabiti, A. Alfonsi, J. Cogliati, D. Mandelli, R. Kinoshita, "RAVEN, a New Software for Dynamic Risk Analysis", in Proceedings for PSAM 12 Conference, Honolulu (USA), 2014

[15] C. J. C. BURGES, "A Tutorial on Support Vector Machines for Pattern Recognition," Data Min. Knowl. Discov., 2, 2, 121–167 (Jun. 1998).

[16] Bentley, J. L. (1975). "Multidimensional binary search trees used for associative searching". Communications of the ACM 18 (9): 509.

[17] Pedregosa et al., "Scikit-learn: Machine Learning in Python", Journal of Machine Learning Research, pp. 2825-2830, 2011.

[18] J. Solis, et al., "Boiling Water Reactor Turbine Trip (TT) Benchmark Volume I: Final Specifications". NEA/NSC/DOC(2001) 1, June (2001).

5. APPENDIX A: Input Files

5.1.1 RELAP-7 nodalization

```
[GlobalParams]
  # these initial values will be used for all the fluid models and will be overrode by local initial values if provided
  # if not provided, these default values will be used
  # scaling factors for flow equations; if not provided, default values will be used
  scaling_factor_var = '1e-3 1e-4 1e-8'
  temperature_sf = '1e-4'
  gravity = '0 0 -9.8'
  global_init_P = 7.e6
  global_init_V = 3.
  global_init_T = 517. #517.252
  model_type = 32
  global_init_alpha = 0.0
  stabilization_type = 'LAPIDUS'
[]
[EoS]
  [./two_phase_eos]
    type = TwoPhaseStiffenedGasEOS
  [../]
  [./vapor_phase_eos]
    type = StiffenedGasEquationOfStateVapor
  [../]
  [./liquid_phase_eos]
    type = StiffenedGasEquationOfStateLiquid
  [../]
  [./eos_nc]
    type = N2Properties
  [../]
[]
 [Materials]
  [./fuel-mat]
    k = 3.7
    Cp = 3.e2
    type = SolidMaterialProperties
    rho = 10.42e3
  [../]
  [./gap-mat]
    k = 0.7
    Cp = 5e3
    type = SolidMaterialProperties
    rho = 1.0
  [../]
  [./clad-mat]
    k = 16
    Cp = 356.
    type = SolidMaterialProperties
    rho = 6.551400e3
  [../]
```

```
[]
[Components]
  #----Main steam line
  # steam venting line
  # ------------------------------------------------
  # water loop to simulate water drawing back to core
  # ------------------------------------------------
  # ------------------------------------------------
  # gas vent loop to simulate venting to dry well
  # ------------------------------------------------
  # ------------------------------------------------
  # wet well
  # ------------------------------------------------
  # separated water return line
  # feed water line
  [./reactor]
    #decay_heat = decayheatcurve
    #decay_heat = decayheatcurveSuperTricked
    initial_power = 3293.0e6
    type = Reactor
  [../]
  [./lowerplenum]
    volume = 61.48
    inputs = 'pipe11(out)'
    center = '0.0  0.0  2.64'
    scale_factors = '1.0E-3  1.0E-9  1.0E-0' # rho, rhoE, vel
    Area = 11.64
    outputs = 'ch1(in)'
    K = '1.0 20'
    initial_T = 517.0
    eos = two_phase_eos
    type = VolumeBranch
  [../]
  [./ch1]
    #length = 3.6576
    #Hw = 5.0e4
    #aw = 2.354927e2
    elem_number_of_hs = '5 1 2' #'5 1 2'
    Ts_init = 517.
    orientation = '0 0 1'
    n_elems = 20 #in relap7 model: 50 200
    power_fraction = '1.0 0.0 0.0'
    Dh = 1.3597E-02
    fuel_type = cylinder
    name_of_hs = 'FUEL GAP CLAD'
    Phf = 18.368e2
    n_heatstruct = 3
    stabilization_type = 'NONE'
    A = 7.8
    material_hs = 'fuel-mat gap-mat clad-mat'
    position = '0 0.0  5.28' #'0 -4.0  5.28'
    PoD = 1.547170E+00
    f = 0.2 #0.05
    type = CoreChannel
    eos = two_phase_eos
    length = 3.66
```

```
  model_type = 32
  HT_geometry_code = 110 # fuel bundle
  width_of_hs = '6.057900e-3  1.524000e-4  9.398000e-4'
  dim_hs = 1
[../]
[./upperplenum]
  volume = 26.99
  inputs = 'ch1(out)'
  center = '0.0  0.0  9.88'
  scale_factors = '1.0E-3  1.0E-8  1.0E-0'
  Area = 14.36
  outputs = 'pipe6(in)'
  K = '3.0 1.0'  #'1.0 1.0'
  initial_T = 517.0
  eos = two_phase_eos
  type = VolumeBranch
[../]
[./pipe6]
  # rising pipe
  A = 3.93  # PI/4 * (0.01)**2
  orientation = '0 0 1'
  Dh = 1.0
  f = 0.1
  Tw = 600 # wall temperature
  Hw = 0. #1e5
  eos = two_phase_eos
  model_type = 32
  length = 2.72
  aw = 400 #
  n_elems = 15 #in relap 7 model :40
  position = '0.0 0.0 10.82'
  type = Pipe
  stabilization_type = 'NONE'
[../]
[./SeparatorDryer]
  volume = 19.30
  inputs = 'pipe6(out)'
  center = '0.0 0.0 14.48'
  scale_factors = '1.0E-3  1.0E-9  1.0E-0'  # rho, rhoE, vel
  Area = 10.27
  outputs = 'pipe7(in)  pipe8(in)'
  K = '1.0  1.0  5.0'
  initial_T = 517.0
  initial_void_fraction = 0.9
  eos = two_phase_eos
  type = SeparatorDryer
[../]
[./pipe7]
  # to steam dome
  A = 3.93
  orientation = '0  0  1'
  Dh = 1.0
  f = 0.1
  Tw = 600
  Hw = 0.0
  eos = vapor_phase_eos
```

```
  model_type = 3
  length = 0.1
  aw = 400.0
  n_elems = 7 # in relap 7 model: 20
  position = '0.0  0.0  15.42'
  type = Pipe
[../]
[./Dome]
  volume = 178.19
  inputs = 'pipe7(out)'
  center = '0.0  0.0  18.92'
  scale_factors = '1.0E-3  1.0E-8  1.0E-0'
  Area = 26.19
  outputs = 'pipe9(in)'
  K = '1.0 1.0'
  eos = vapor_phase_eos
  type = VolumeBranch
[../]
[./pipe9]
  # main steam line coming out of dome
  A = 1.32
  orientation = '0 1 0'
  Dh = 1.0
  f = 0.1
  Tw = 600
  Hw = 0.0
  eos = vapor_phase_eos
  model_type = 3
  length = 1.0
  aw = 400.0
  n_elems = 3 # in relap 7 model: 5
  position = '0.0  3  18.92'
  type = Pipe
[../]
[./SteamLineBranch]
  volume = 2.64 #1.32
  inputs = 'pipe9(out)'
  center = '0.0  4  18.92'
  scale_factors = '1.0E-4 1.0E-8 1.0'
  Area = 1.32
  outputs = 'pipe14(in) pipe_venting1(in)'
  K = '0.0 0.0  0'
  initial_T = 517.0
  eos = vapor_phase_eos
  type = VolumeBranch
[../]
[./pipe14]
  # main steam line to MIV
  A = 1.32
  orientation = '0 1 0'
  Dh = 1.0
  f = 0.0
  Tw = 600
  Hw = 0.0
  eos = vapor_phase_eos
  model_type = 3
```

```
  length = 1.0
  aw = 400.0
  n_elems = 3 # in relap 7 model: 5
  position = '0.0  4  18.92'
  type = Pipe
[../]
[./MainIsolationValve]
  volume = 1.32
  inputs = 'pipe14(out)'
  center = '0.0  5.0  18.92'
  scale_factors = '1.0E-4  1.0E-11'  # rho, rhoE
  Area = 1.32
  outputs = 'pipe_steam_turbine(in)'
  K = '0.0 0.0'
  initial_T = 517.0
  initial_status = open
  eos = vapor_phase_eos
  trigger_time = 1 #1.0E5
  type = Valve
  response_time = 10 #1.1E5
[../]
[./pipe_steam_turbine]
  # main steam line to TDV
  A = 1.32
  orientation = '0 1 0'
  Dh = 1.0
  f = 0.0
  Tw = 600
  Hw = 0.0
  eos = vapor_phase_eos
  model_type = 3
  length = 1.0
  aw = 400.0
  n_elems = 3 # in relap 7 model: 5
  position = '0.0  5  18.92'
  type = Pipe
[../]
[./outlet1]
  weak_bc = false
  T_bc = 517
  p_bc = 7.0e6
  eos = vapor_phase_eos
  input = 'pipe_steam_turbine(out)'
  type = TimeDependentVolume
[../]
[./pipe_venting1]
  #stabilization_type = 'NONE'
  # geometry
  A = 1.2566e-1
  orientation = '0 0 -1'
  Dh = 0.4
  f = 0.1
  initial_P = 7e6
  initial_T = 517.
  Hw = 0.0 # not setting Hw means that Hw is calculated by models, need set 0 for no heat transfer
  initial_V = 0.
```

```
  eos = vapor_phase_eos
  n_elems = 4 # in relap 7 model: 10
  length = 5 #0.1
  model_type = 3
  position = '0  4  18.92'
  type = Pipe
[../]
[./branch_venting1]
  volume = 2.5132e-1
  inputs = 'pipe_venting1(out)'
  center = '0.0  4  13.92'
  scale_factors = '1.0E-4 1.0E-8 1.0'
  Area = 1.2566e-1
  outputs = 'pipe_venting2(in)'
  K = '100 100'
  initial_T = 517.0
  eos = vapor_phase_eos
  type = VolumeBranch
[../]
[./pipe_venting2]
  #stabilization_type = 'NONE'
  # geometry
  A = 1.2566e-1
  orientation = '0 1 0'
  Dh = 0.4
  f = 0.1
  initial_P = 7e6
  initial_T = 517.
  Hw = 0.0 # not setting Hw means that Hw is calculated by models, need set 0 for no heat transfer
  initial_V = 0.
  eos = vapor_phase_eos
  n_elems = 4 # in relap 7 model: 10
  length = 4 #0.1
  model_type = 3
  position = '0  4  13.92'
  type = Pipe
[../]
[./branch_venting2]
  volume = 2.5132e-1
  inputs = 'pipe_venting2(out)'
  center = '0.0  8  13.92'
  scale_factors = '1.0E-4 1.0E-8 1.0'
  Area = 1.2566e-1
  outputs = 'pipe_turbine_inlet(in)'
  K = '100 100'
  initial_T = 517.0
  eos = vapor_phase_eos
  type = VolumeBranch
[../]
[./pipe_turbine_inlet]
  #stabilization_type = 'NONE'
  # geometry
  A = 1.2566e-1
  orientation = '0 0 -1'
  Dh = 0.4
  f = 0.1
```

```
  initial_P = 7e6
  initial_T = 517.
  Hw = 0.0 # not setting Hw means that Hw is calculated by models, need set 0 for no heat transfer
  initial_V = 0.
  eos = vapor_phase_eos
  n_elems = 4 # in relap 7 model: 10
  length = 16.92
  model_type = 3
  position = '0  8  13.92'
  type = Pipe
[../]
[./turbine]
  inputs = 'pipe_turbine_inlet(out)'
  p0_design = 7e6 #6e6
  scale_factors = '1e-1 1e-1 1e-5 1e-7' # for inlet pressure, outlet pressure, outlet density, and shaft work
  Initial_p = 7e6
  outputs = 'pipe_turbine_outlet(in)'
  Initial_T = 517.
  T0_design = 517
  eos = vapor_phase_eos
  Turbine_efficiency = 0.6 #0.9
  is_shutdown = 'false' #'true'
  relative_mass_flow_rate_design = 0.8
  pressure_ratio_design = 3. #3.0
  type = Turbine
  max_mass_flow_rate = 2e-1  # steady state design point
[../]
[./pipe_turbine_outlet]
  #stabilization_type = 'SUPG'
  #eos = two_phase_eos
  # geometry
  #initial_void_fraction = 1.0
  A = 1.2566e-1 #7.854e-1
  orientation = '0 0 -1'
  Dh = 0.4 #1
  f = 0.1 #10
  initial_P = 1.5e5 #1e5
  initial_T = 400 #300.
  Hw = 0.0
  initial_V = 1e-2 #0.
  eos = vapor_phase_eos
  n_elems = 4 # in relap 7 model: 10
  length = 6.5 #0.1
  model_type = 3
  position = '0  9   -3' #'0  4.1   18.82'
  type = Pipe
[../]
[./pipe_RCIC_pump_inlet]
  # geometry
  A = 3.141593e-2
  orientation = '0  0  1'
  Dh = 0.2
  f = 0.001 #0.2
  initial_P = 1e5
  initial_T = 300
  Hw = 0.0 # not setting Hw means that Hw is calculated by models, need set 0 for no heat transfer
```

```
    initial_V = 0 #1e-3
    eos = liquid_phase_eos
    n_elems = 7 # in relap 7 model: 20
    length = 8
    model_type = 3
    position = '0  6  -11'
    type = Pipe
    stabilization_type = 'SUPG'
  [../]
  [./RCIC_pump]
    # now no-used but still required parameters, give them some whatever values
    #Area = 0.007853982
    inputs = 'pipe_RCIC_pump_inlet(out)'
    outputs = 'pipe_RCIC_pump_outlet(in)'
    Initial_pressure = 1.0e5
    eos = liquid_phase_eos
    mass_flow_rate = 2e-1   # this number should be equal or smaller than the  max_mass_flow_rate for
turbine
    type = IdealPump   # IdealPump is good to simulate closed valve for incompressible fluid
  [../]
  [./pipe_RCIC_pump_outlet]
    # geometry
    A = 3.141593e-2
    orientation = '0 -1 0'
    Dh = 0.2
    f = 1e-3
    initial_P = 7e6
    initial_T = 517
    Hw = 0.0 # not setting Hw means that Hw is calculated by models, need set 0 for no heat transfer
    initial_V = 0 #1e-3
    eos = liquid_phase_eos
    n_elems = 3 # in relap 7 model: 5
    length = 1
    model_type = 3
    position = '0  5  -3'
    type = Pipe
    stabilization_type = 'SUPG'
  [../]
  [./branch_RCIC_water_line]
    volume = 0.007853982
    inputs = 'pipe_RCIC_pump_outlet(out)'
    center = '0  4  -2'
    scale_factors = '1.0E-4 1.0E-8 1.0'
    Area = 3.141593e-2
    outputs = 'pipe_RCIC_to_feedwater_line(in)'
    K = '0 0'
    initial_T = 517.0
    eos = liquid_phase_eos
    type = VolumeBranch
  [../]
  [./pipe_RCIC_to_feedwater_line]
    # geometry
    A = 3.141593e-2
    orientation = '0 0 1'
    Dh = 0.2
    f = 1e-4 #0.01
```

```
  initial_P = 7e6
  initial_T = 517
  Hw = 0.0 # not setting Hw means that Hw is calculated by models, need set 0 for no heat transfer
  initial_V = 0 #1e-3
  eos = liquid_phase_eos
  n_elems = 9 # in relap 7 model: 30
  length = 15.52
  model_type = 3
  position = '0  4  -3'
  type = Pipe
  stabilization_type = 'SUPG'
[../]
[./pipe_to_dry_well]
  # geometry
  A = 0.031415927
  orientation = '0 1 0'
  Dh = 0.2
  f = 0.01 #0.2
  initial_P = 1e5
  initial_T = 300
  Hw = 0.0 # not setting Hw means that Hw is calculated by models, need set 0 for no heat transfer
  initial_V = 0
  eos = eos_nc
  n_elems = 15 # in relap 7 model: 50
  length = 0.2
  model_type = 3
  position = '0  11  -4'
  type = Pipe
  stabilization_type = 'NONE'
[../]
[./VacuumBreaker]
  volume = 3.142e-3
  inputs = 'pipe_to_dry_well(out)'
  center = '0  11.2  -4'
  scale_factors = '1.0E-4  1.0E-11'  # rho, rhoE
  Area = 0.031415927
  outputs = 'pipe_to_dry_well2(in)'
  K = '0.0 0.0'
  initial_T = 300
  initial_status = close
  eos = eos_nc
  Initial_pressure = 1.e5
  trigger_time = 1.0E100
  type = Valve
  response_time = 1.1E100
[../]
[./pipe_to_dry_well2]
  # geometry
  A = 0.031415927
  orientation = '0 1 0'
  Dh = 0.2
  f = 0.01 #0.2
  initial_P = 1e5
  initial_T = 300
  Hw = 0.0 # not setting Hw means that Hw is calculated by models, need set 0 for no heat transfer
  initial_V = 0
```

```
  eos = eos_nc
  n_elems = 15 # in relap 7 model:  50
  length = 0.2
  model_type = 3
  position = '0  11.3  -4'
  type = Pipe
  stabilization_type = 'NONE'
[../]
[./Dry_well]
  eos = eos_nc
  input = 'pipe_to_dry_well2(out)'
  p_bc = 1.e5
  type = TimeDependentVolume
  T_bc = 300
[../]
[./wet_well]
  eos_water = liquid_phase_eos
  K_or = 1.0
  K_ir = 1e6 #0.5
  K_i = 100 #1.0
  Lt = 8
  K_o = 0.1 #0.5
  cooling_rate = 0.0
  scale_factors = '1e-5 1e-10 1e-7 1e-13 1e-6' # for mg, me_g, mw, me_w, Lw
  Lw_initial = 4 #5
  type = WetWell
  inputs = 'pipe_turbine_outlet(out)'
  z_in = 2.5
  eos_vapor = vapor_phase_eos
  outputs = 'pipe_RCIC_pump_inlet(in) pipe_to_dry_well(in)'
  K_vr = 1.0
  alpha_s = 1e3
  K_v = 0.5
  Ac = 892.5 FIXME
  p_gas_initial = 1.e5
  z_out = 1
  T_initial = 300.0
  eos_nc_gas = eos_nc
[../]
[./pipe8]
  # discharge water line from SeparatorDryer
  A = 3.93
  orientation = '0  1  0'
  Dh = 1.0
  f = 0.1
  Tw = 600
  Hw = 0.0
  eos = liquid_phase_eos
  model_type = 3
  length = 0.5 #2.0
  aw = 400.0
  n_elems = 3 # in relap 7 model: 5
  position = '0.0  2.0  14.48'
  type = Pipe
[../]
[./DownComer]
```

```
  volume = 201.3 #171.3
  inputs = 'pipe8(out) pipe_feedwater3(out)'
  center = '0.0  2.75  9.81' #'0.0  4.0  9.81'
  scale_factors = '1.0E-4  1.0E-10  1.0E-2'  # mass, energy, and level
  dome_eos = vapor_phase_eos
  outputs = 'pipe10(in)'
  K = '1.0 10.0 1.0'
  Area = 15
  initial_T = 517.0
  initial_level = 13.42 #11.42
  eos = liquid_phase_eos
  display_pps = 'true'
  dome_component = 'Dome'
  type = DownComer
[../]
[./pipe10]
  # downcomer pipe
  #position = '0.0  5.0  10.51'
  #length = 6.42
  A = 8.55
  orientation = '0  0  -1'
  Dh = 1.0
  f = 0.1
  Tw = 600
  Hw = 0.0
  eos = liquid_phase_eos
  model_type = 3
  length = 0.5
  aw = 400.0
  n_elems = 3 # in relap 7 model: 5
  position = '0.0  2.75  2.10' #'0.0  5.0  4.10'
  type = Pipe
  stabilization_type = 'SUPG'
[../]
[./Pump]
  #type = IdealPump
  #mass_flow_rate = 12915.0
  inputs = 'pipe10(out)'
  Head = 40 #33.0
  Area = 3.0
  outputs = 'pipe11(in)'
  eos = liquid_phase_eos
  Initial_pressure = 7.3e6
  K_reverse = '10.  10.'
  type = Pump
[../]
[./pipe11]
  # pipe to lower plenum
  A = 8.55
  orientation = '0  0  -1'
  Dh = 1.0
  f = 0.1
  Tw = 600.0
  Hw = 0.0
  eos = liquid_phase_eos
  model_type = 3
```

```
  length = 0.5
  aw = 400.0
  n_elems = 3 # in relap 7 model: 5
  position = '0.0  2.75  1.60' #'0.0  5.0  3.60'
  type = Pipe
  stabilization_type = 'SUPG'
[../]
[./inlet]
  #    type = TDM
  #    massflowrate_bc = 1909.2
  T_bc = 508.
  p_bc = 7.1e6
  eos = liquid_phase_eos
  void_fraction_bc = -0.01
  input = 'pipe_feedwater1(in)'
  type = TimeDependentVolume
[../]
[./pipe_feedwater1]
  #feedwater line from TDV
  #    f = 0.01
  #stabilization_type = 'LAPIDUS'
  A = 1.32
  orientation = '0  -1  0'
  Dh = 1.0
  f = 0.01 #1
  Tw = 600.0
  Hw = 0.
  eos = liquid_phase_eos
  model_type = 3
  length = 1.0
  aw = 400.0
  n_elems = 3 # in relap 7 model: 5
  position = '0.0  6.0  12.52' #'0.0  7.0  12.52'
  type = Pipe
  stabilization_type = 'SUPG'
[../]
[./FeedWaterValve]
  volume = 1.32
  inputs = 'pipe_feedwater1(out)'
  center = '0.0  5.0  12.52'
  scale_factors = '1.0E-4  1.0E-11'  # rho, rhoE
  Area = 1.32
  outputs = 'pipe_feedwater2(in)'
  K = '0.0 0.0'
  initial_T = 517.0
  initial_status = open
  eos = liquid_phase_eos
  trigger_time = 1 #1.0E5
  type = Valve
  response_time = 1 #1.1E5
[../]
[./pipe_feedwater2]
  #feedwater line from feed water valve
  #    f = 0.01
  #stabilization_type = 'LAPIDUS'
  A = 1.32
```

```
    orientation = '0  -1  0'
    Dh = 1.0
    f = 0.01 #1
    Tw = 600.0
    Hw = 0.
    eos = liquid_phase_eos
    model_type = 3
    length = 1.0
    aw = 400.0
    n_elems = 3 # in relap 7 model: 5
    position = '0.0  5.0  12.52' #'0.0  7.0  12.52'
    type = Pipe
    stabilization_type = 'SUPG'
  [../]
  [./branch_feedwater_line]
    volume = 1.32
    inputs = 'pipe_feedwater2(out)'
    center = '0.0  4.0  12.52'
    scale_factors = '1.0E-4 1.0E-8 1.0'
    Area = 1.32
    outputs = 'pipe_feedwater3(in)  pipe_RCIC_to_feedwater_line(out)'
    K = '0  0  0'
    initial_T = 517.0
    eos = liquid_phase_eos
    type = VolumeBranch
  [../]
  [./pipe_feedwater3]
    #feedwater line to downcomer
    #f = 0
    #stabilization_type = 'LAPIDUS'
    A = 1.32
    orientation = '0  -1  0'
    Dh = 1.0
    f = 0.01
    Tw = 600.0
    Hw = 0.
    eos = liquid_phase_eos
    model_type = 3
    length = 1.0
    aw = 400.0
    n_elems = 3 # in relap 7 model: 5
    position = '0.0  4.0  12.52' #'0.0  6.0  12.52'
    type = Pipe
    stabilization_type = 'SUPG'
  [../]
[]
[Postprocessors]
  [./core_void]
    variable = void_fraction_HEM
    type = ElementAverageValue
    block = 'ch1:pipe'
  [../]
[]
[Preconditioning]
  # active = 'FDP_PJFNK'
  #active = 'FDP_Newton'
```

```
  active = 'SMP_PJFNK'
  [./SMP_PJFNK]
    petsc_options_iname = '-mat_fd_type -mat_mffd_type'
    petsc_options_value = 'ds ds'
    full = true
    type = SMP
    solve_type = 'PJFNK'
  [../]
  [./FDP_PJFNK]
    petsc_options_iname = '-mat_fd_type  -mat_mffd_type'
    petsc_options_value = 'ds            ds'
    full = true
    type = FDP
    solve_type = 'PJFNK'
  [../]
  [./FDP_Newton]
    petsc_options_iname = '-mat_fd_coloring_err'
    petsc_options_value = '1.e-10'
    full = true
    type = FDP
    solve_type = 'NEWTON'
  [../]
[]
[Executioner]
  #type = Transient
  #predictor_scale = 0.5
  #
  #  [./TimeStepper]
  #    type = SolutionTimeAdaptiveDT
  #    dt = 0.5
  #    percent_change = 0.15
  #  [../]
  #  num_steps = 5000000000 The number of timesteps in a transient run
  #restart_file_base = SBO_raven_8_11_out_restart_12680
  #restart_file_base = SBO_8_11_small_steady_out_cp/0572
  nl_abs_tol = 1e-4 #5e-5
  petsc_options_value = '30 lu'
  nl_max_its = 20 #11 #15
  restart_file_base = 0750
  type = RavenExecutioner
  start_time = 0 #1834.0600
  nl_rel_tol = 1e-8 #1e-9
  dump_raven_init = false
  l_tol = 1e-4 #1e-6   Relative linear tolerance for each Krylov solve
  dtmin = 1.e-9
  dt = 1e-2
  scheme = 'implicit-euler' # this is not default option anymore
  petsc_options_iname = '-ksp_gmres_restart -pc_type'
  l_max_its = 30 #60 Number of linear iterations for each Krylov solve
  end_time = 450
  [./TimeStepper]
    # steady state time step control
    #time_t  = '0      0.01     0.1    0.5     20      50    100  1e5'
    #time_dt = '1e-3    2.e-3    2.e-3  1.e-2   1.1e-2  1.5e-2  2e-2 2e-1'
    #time_dt = '5e-3    5.e-3    5.e-3  2.e-2   4.e-2  1e-1  2e-1   5e-1'
    #  transient time step control
```

```
    time_t = '0      0.01     0.1    0.5     20      50     100  200  298 1820  1e5'
    time_dt = '1e-2    2.e-2     2.e-2  1.e-1   1e-1    2e-1   1e-1 2e-1 6e-2 3e-1  1e-1'
    type = FunctionDT
  [../]
  [./Quadrature]
    # Specify the order as FIRST, otherwise you will get warnings in DEBUG mode...
    type = TRAP
    order = FIRST
  [../]
[]
[Outputs]
  # Turn on performance logging
  #interval = 20
  #  num_checkpoint_files = 1
  exodus = false
  output_intermediate = false
  output_displaced = false
  output_initial = false
  perf_log = true
  csv = true
  [./console]
    perf_log = false
    type = Console
  [../]
[]
[Debug]
  #  show_var_residual_norms = true
[]
[Controlled]
  control_logic_input = SBO_control_logic_final
  [./turbine_max_mass_flow_rate]
    print_csv = true
    data_type = double
    property_name = max_mass_flow_rate
    component_name = turbine
  [../]
  [./RCIC_pump_flow_rate]
    print_csv = true
    data_type = double
    property_name = mass_flow_rate
    component_name = RCIC_pump
  [../]
  [./HeadPump]
    print_csv = true
    property_name = Head
    data_type = double
    component_name = Pump
  [../]
  [./ReactorPowerFract]
    print_csv = true
    property_name = FUEL:power_fraction
    data_type = double
    component_name = ch1
  [../]
  [./ReactorPower]
    print_csv = false
```

```
    property_name = initial_power
    data_type = double
    component_name = reactor
  [../]
[]
[RavenAuxiliary]
  [./initialPowerFractionLevel]
    print_csv = true
    data_type = double
    initial_value = 0.0
  [../]
  [./RealPower]
    print_csv = true
    data_type = double
    initial_value = 0.0
  [../]
  [./initialHeadPump]
    print_csv = true
    data_type = double
    initial_value = 40.0
  [../]
  [./DGsFailTime]
    print_csv = true
    data_type = double
    initial_value = 0.0
  [../]
  [./DGsRecoveryTime]
    print_csv = true
    data_type = double
    initial_value = 0.0
  [../]
  [./CollapsedTimeParameter]
    print_csv = true
    data_type = double
    initial_value = 0.0
  [../]
  [./CanDGsFailAgain]
    print_csv = false
    data_type = bool
    initial_value = false
  [../]
  [./AuxSystemAvailable]
    print_csv = true
    data_type = bool
    initial_value = false
  [../]
  [./OffSitePowerRecoveryTime]
    print_csv = true
    data_type = double
    initial_value = 0.0
  [../]
  [./BurnUp]
    print_csv = true
    data_type = double
    initial_value = 0.0
  [../]
```

```
  [./PbThreshold2Ddist]
    print_csv = true
    data_type = double
    initial_value = 0.0
  [../]
  [./CladTempFailure]
    print_csv = true
    data_type = double
    initial_value = 0.0
  [../]
  [./CladFailed]
    print_csv = true
    data_type = bool
    initial_value = false
  [../]
  [./MonitorProbabilityLevel]
    print_csv = true
    data_type = double
    initial_value = 0.0
  [../]
  [./sboStartTime]
    print_csv = true
    data_type = double
    initial_value = 1.0
  [../]
  [./keepGoing]
    print_csv = true
    data_type = bool
    initial_value = true
  [../]
  [./Pdf2Ddistribution]
    print_csv = true
    data_type = double
    initial_value = 0.0
  [../]
  [./PdfDGs]
    print_csv = true
    data_type = double
    initial_value = 0.0
  [../]
  [./PdfOffSite]
    print_csv = true
    data_type = double
    initial_value = 0.0
  [../]
  [./PdfBU]
    print_csv = true
    data_type = double
    initial_value = 0.0
  [../]
  [./PointProbability]
    print_csv = true
    data_type = double
    initial_value = 0.0
  [../]
[]
```

```
[RavenTools]
  [./PumpCoastDown]
    coefficient = 3 #2
    initial_flow_rate = 40
    type = pumpCoastdownExponential
  [../]
  [./DecayHeat]
    fitting_type = linear
    type = TableFunction
    y_coordinates = '1 0.8 0.636236373 0.636204562 0.636172753 0.636140945 0.636109139
0.636077334 0.636045531 0.636013729 0.63598193 0.635950131 0.635918335 0.635886539
0.635854746 0.635822954 0.635791164 0.635759375 0.635727588 0.635695802 0.635664018
0.635505122 0.635346265 0.635187449 0.635028672 0.634711237 0.63439396 0.634076843
0.633759884 0.633126441 0.632493631 0.631861453 0.631229907 0.630598993 0.629968709
0.626826727 0.623700416 0.620589697 0.617494493 0.28590802'
    x_coordinates = '0 1 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 125 150 175 200 250
300 350 400 500 600 700 800 900 1000 1500 2000 2500 3000 80000'

  [../]
  [./PumpRampUp]
    fitting_type = linear
    y_coordinates = '0  1.0'
    type = TableFunction
    x_coordinates = '0  5.0'
  [../]
[]
[Distributions]
  # CladFailure2DTriangularNormal is the distribution that drives the plant failure
   [./CladFailure2DTriangularNormal]
      alpha = '0 0'
      beta  = '0 0'
      function_type = CDF
      type = MultiDimensionalCartesianSpline
      data_filename = TriangularExponential2D.txt
    [../]
  # BurnUpDistribution is the distribution that define the sampling on the II dimension of
CladFailure2DTriangularNormal
  # This distribution is needed to create the 2D grid sampling (it define the Probability Thresholds)
[./DGsRecoveryTime]
    k = 0.745
    V_window_Up = 237.17909
    V_window_Low = 236.7169
    xMax = 600.0
    xMin = 1.0
    type = WeibullDistribution
    lambda = 120.0
  [../]
  [./OffSitePowerRecoveryTime]
    type = LogNormalDistribution
    V_window_Up = 99.66034
    mu = 2.66
    V_window_Low = 99.21281
    xMax = 600.0
    xMin = 20.0
    sigma = 2.0
  [../]
```

```
  [./DummyPbForThresholdSet]
    xMin = 0.0
    type = UniformDistribution
    V_window_Low = 0.3194424
    V_window_Up = 0.320214
    xMax = 1.0
  [../]
  [./BurnUpDistribution]
    V_window_Up = 21.94146
    V_window_Low = 21.89517
    xMax = 60.0
    xMin = 0.0
    type = ExponentialDistribution
    lambda = 0.01
  [../]
[]
[Monitored]
  [./ch1cladtemperature]
    operator = NodalMaxValue #ElementAverageValue
    path = CLAD:TEMPERATURE
    data_type = double
    component_name = ch1
  [../]
    []
```

5.1.2 RAVEN control Logic

```
import sys
import math
coordinate = None
triangularBu = None

def restart_function(monitored, controlled, auxiliary):

    indebugmode = False
    global coordinate
    global triangularBu
    if coordinate is None:
      coordinate = distribution1D.vectord_cxx(2)
      coordinate[0]=0.0
      coordinate[1]=0.0
    # initial pump head
    auxiliary.initialHeadPump = 40.0
    # get sampled vars (these calls are the same for all the sampling strategy types (except DET))
    if not indebugmode:
      auxiliary.DGsRecoveryTime         = distributions.DGsRecoveryTime.getDistributionRandom()
      auxiliary.OffSitePowerRecoveryTime                                                    =
distributions.OffSitePowerRecoveryTime.getDistributionRandom()
      auxiliary.BurnUp                  = distributions.BurnUpDistribution.getDistributionRandom()
      auxiliary.PbThreshold2Ddist       = distributions.DummyPbForThresholdSet.getDistributionRandom()
    else:
      auxiliary.DGsRecoveryTime         = distributions.DGsRecoveryTime.getDistributionRandom()
      auxiliary.OffSitePowerRecoveryTime                                                    =
distributions.OffSitePowerRecoveryTime.getDistributionRandom()
      auxiliary.BurnUp                  = distributions.BurnUpDistribution.getDistributionRandom()
      auxiliary.PbThreshold2Ddist       = distributions.DummyPbForThresholdSet.getDistributionRandom()
```

```
    # compute the key times
    auxiliary.CollapsedTimeParameter                                                    =
min(auxiliary.DGsRecoveryTime,auxiliary.OffSitePowerRecoveryTime)
    auxiliary.initialPowerFractionLevel= controlled.ReactorPowerFract
    auxiliary.RealPower = controlled.ReactorPower
    print('DGsRecoveryTime: ',        str(auxiliary.DGsRecoveryTime))
    print('OffSitePowerRecoveryTime: ',str(auxiliary.OffSitePowerRecoveryTime))
    print('BurnUp: ',                 str(auxiliary.BurnUp))
    print('PbThreshold2Ddist: ',      str(auxiliary.PbThreshold2Ddist))
    triangularBu                       =                  distribution1D.BasicTriangularDistribution(1477.59-
math.exp(0.092354*auxiliary.BurnUp),1255.3722-math.exp(0.092354*auxiliary.BurnUp),1699.8167-
math.exp(0.092354*auxiliary.BurnUp))
    print("CREATED DISTRIBUTION ON THE FLY...LOW:")
    print(str(1255.3722-math.exp(-0.092354*auxiliary.BurnUp)))
    print("PEAK:")
    print(str(1477.59-math.exp(-0.092354*auxiliary.BurnUp)))
    print("UP:")
    print(str(1699.8167-math.exp(-0.092354*auxiliary.BurnUp)))
def control_function(monitored, controlled,auxiliary):
    global coordinate
    global triangularBu
    if auxiliary.CladFailed and monitored.time_step > 3:
      auxiliary.keepGoing = False
      print('CLAD FAILED')
      return
    coordinate[0] = monitored.ch1cladtemperature
    coordinate[1] = auxiliary.BurnUp
    print("LoweBound  2D Dist: " + str(1255.3722-math.exp(0.092354*auxiliary.BurnUp)))
    print("Peak       2D Dist: " + str(1477.59-math.exp(0.092354*auxiliary.BurnUp)))
    print("UpperBound 2D Dist: " + str(1699.8167-math.exp(0.092354*auxiliary.BurnUp)))
    print("CDF of 2D ",str( triangularBu.Cdf(monitored.ch1cladtemperature)))
    print("PDF of 2D ",str( triangularBu.Pdf(monitored.ch1cladtemperature)))
    #auxiliary.MonitorProbabilityLevel = distributions.CladFailure2DTriangularNormal.Cdf(coordinate)
    auxiliary.MonitorProbabilityLevel = triangularBu.Cdf(monitored.ch1cladtemperature)
    auxiliary.Pdf2Ddistribution       = triangularBu.Pdf(monitored.ch1cladtemperature)
    auxiliary.PdfDGs                  = distributions.DGsRecoveryTime.Pdf(auxiliary.DGsRecoveryTime)
    auxiliary.PdfOffSite                                                                =
distributions.OffSitePowerRecoveryTime.Pdf(auxiliary.OffSitePowerRecoveryTime)
    auxiliary.PdfBU                   = distributions.BurnUpDistribution.Pdf(auxiliary.BurnUp)
    auxiliary.PointProbability        = auxiliary.PdfDGs*auxiliary.Pdf2Ddistribution*auxiliary.PdfOffSite
    # pump head control
    if (monitored.time <  auxiliary.sboStartTime)                                       : controlled.HeadPump =
auxiliary.initialHeadPump
    if (monitored.time >= (auxiliary.sboStartTime)) and (not auxiliary.AuxSystemAvailable):
      controlled.HeadPump = tools.PumpCoastDown.compute(monitored.time-auxiliary.sboStartTime)
      controlled.turbine_max_mass_flow_rate    =    min(0.2    +    (auxiliary.initialHeadPump    -    0.2)    *
(monitored.time) / 10, auxiliary.initialHeadPump)
      controlled.RCIC_pump_flow_rate = min((0.2 + (auxiliary.initialHeadPump - 0.2) * (monitored.time) /
10),auxiliary.initialHeadPump)
    if controlled.HeadPump < auxiliary.initialHeadPump*1.0e-5                           : controlled.HeadPump =
auxiliary.initialHeadPump*1.0e-5
    #
    if (monitored.time >= auxiliary.sboStartTime):
      print('SBO CONDITION')
      # scram => decay heat curve
      controlled.ReactorPowerFract                                                      =
```

```
auxiliary.initialPowerFractionLevel*tools.DecayHeat.compute(monitored.time-auxiliary.sboStartTime)
    auxiliary.RealPower = controlled.ReactorPower*controlled.ReactorPowerFract
    # check if aux cooling system is operative again
    if monitored.time > (auxiliary.sboStartTime + auxiliary.CollapsedTimeParameter):
auxiliary.AuxSystemAvailable = True
    else                                                    : auxiliary.AuxSystemAvailable = False
    if auxiliary.AuxSystemAvailable:
      print('COOLING SYSTEM UP')
      # the cooling is guaranteed
      if controlled.HeadPump <= auxiliary.initialHeadPump*0.5:
      if (monitored.time - (auxiliary.sboStartTime + auxiliary.CollapsedTimeParameter)) < 5.0:
          actualPumpHead = controlled.HeadPump
          controlled.HeadPump = tools.PumpRampUp.compute(monitored.time - (auxiliary.sboStartTime +
auxiliary.CollapsedTimeParameter))*auxiliary.initialHeadPump
          if controlled.HeadPump < actualPumpHead:controlled.HeadPump = actualPumpHead
        else:
          controlled.HeadPump = auxiliary.initialHeadPump*0.5 #auxiliary.initialHeadPump*0.50
          controlled.turbine_max_mass_flow_rate = auxiliary.initialHeadPump
          controlled.RCIC_pump_flow_rate = auxiliary.initialHeadPump
      controlled.ReactorPowerFract = auxiliary.initialPowerFractionLevel*0.01
  if triangularBu.Cdf(monitored.ch1cladtemperature) >= auxiliary.PbThreshold2Ddist:
    auxiliary.CladFailed = True
  if auxiliary.CladFailed: auxiliary.CladTempFailure = monitored.ch1cladtemperature
  return

def keep_going_function(monitored, controlled, auxiliary):
  return auxiliary.keepGoing
```