INL/EXT-15-36100

Light Water Reactor Sustainability Program

Improving Limit Surface Search Algorithms in RAVEN Using Acceleration Schemes

July 2015

DOE Office of Nuclear Energy

INL/EXT-15-36100

Light Water Reactor Sustainability Program

Improving Limit Surface Search Algorithms in RAVEN Using Acceleration Schemes

Andrea Alfonsi, Cristian Rabiti, Diego Mandelli, Joshua Cogliati, Sonat Sen, Curtis Smith

July 2015

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov/lwrs

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

EXECUTIVE SUMMARY

The RAVEN code is becoming a comprehensive tool to perform probabilistic risk assessment, uncertainty quantification, and verification and validation. The RAVEN code is being developed to support the risk-informed safety margin characterization pathway to provide an set of methodologies and algorithms for advanced risk analysis.

The risk-informed safety margin characterization approach applies stochastic analysis tools to system simulator codes. The fundamental idea behind this coupling approach is to perturb (by employing sampling strategies) timing and sequencing of events, internal parameters of the system codes (i.e., uncertain parameters of the physics model), and initial conditions to estimate values ranges and associated probabilities of figures of merits of interest for engineering and safety (e.g., core damage probability). This approach applied to complex systems such as nuclear power plants requires performing a series of computationally expensive simulation runs. The large computational burden is caused by the large set of (uncertain) parameters characterizing those systems. Consequently, exploring the uncertain/parametric domain, with a good level of confidence, is generally not affordable, within the limited computational resources that are currently available. In addition, the recent tendency to develop newer tools, characterized by higher accuracy and needs for larger computational resources (if compared with the presently-used legacy codes that were developed decades ago), has made this issue even more compelling. To overcome these limitations, the strategy for exploration of the uncertain/parametric space needs to use, at best, resources focusing the computational effort in those regions of the uncertain/parametric space that are “interesting” (e.g., risk-significant regions of the input space) with respect to the targeted figures of merit (for example, the failure of the system, subject of the analysis). These methodologies are named, in the RAVEN environment, adaptive sampling strategies. These methodologies infer the overall system response from surrogate models, constructed from already existing samples (produced using high-fidelity simulations), and suggest the most relevant location (coordinate in the input space) of the next sampling point to be explored in the uncertain/parametric domain. When using those methodologies, it is possible to understand features of the system response with a small number of carefully selected samples.

This report focuses on development and improvement of the limit surface (LS) search. The LS is an important concept in system reliability analysis. The LS could be briefly described as a hyper-surface in the system uncertainty/parametric space separating the regions leading to a prescribed system outcome. For example, if the uncertainty/parametric space is the one generated by the reactor power level and duration of the batteries, the system is a nuclear power plant and the system outcome discriminating variable is the clad failure in a station black-out scenario, then the LS separates the combinations of reactor power level and battery duration that lead to clad failure from those that does not.

CONTENTS

FIGURES

TABLES

ACRONYMS

API	application program interface
CDF	cumulative distribution function
CPU	central processing unit
DET	dynamic event tree
DG	diesel generator
ECCS	emergency core cooling system
FOM	figure of merit
LS	limit surface
MC	Monte-Carlo
N-D	N-dimensional
NPP	nuclear power plant
PDF	probability distribution function
PRA	probabilistic risk assessment
PWR	pressurized water reactor
RISMC	risk-informed safety margin characterization
ROM	reduced order model
SBO	station black-out
SVM	support vector machine

Improving Limit Surface Search Algorithms in RAVEN Using Acceleration Schemes

1. INTRODUCTION

RAVEN [1–6] is advancing its capability to perform statistical analyses of stochastic dynamic systems, putting a big effort in the identification and development of methodologies able to identify the region of interest in the uncertain/parametric space optimizing the computational resources. This effort is aligned with RAVEN's mission to provide the tools needed by the risk-informed safety margin characterization (RISMC) path-lead [7] under the Department of Energy Light Water Reactor Sustainability Program. [8]

Investigation of the probabilistic evolution of accident scenarios for a complex system such as a nuclear power plant (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 central processing unit [CPU] hours). At the same time, the probabilistic analysis requires thousands of runs (simulation of one of the possible scenario evolutions) to investigate outcomes characterized by low probability and severe consequence.

The probabilistic analysis is performed by (1) sampling the stochastic parameters, and (2) evaluating the system response for the given set of sampled parameters. As already mentioned, sampling the uncertain domain generally requires a large amount of samples, increasing with non-linearity of the physics model representing the figure of merit (FOM) of interest, and decreasing with the probability of the event under consideration (low probability events require a large number of samples). In addition, it is worth mentioning that the scope of the analysis is not only to determine outcome variable values or probabilities such as core damage probability but more in general, to evaluate the overall system response for different combinations of the stochastic parameters (e.g., response surfaces).

The large number of samples required and computational cost of each sample (single stochastic realization) may limit the capability to perform a full probabilistic risk assessment (PRA) analysis of complex systems. To effectively answer this challenge, it is necessary to find approaches to reduce the (1) number of samples needed to perform a comprehensive PRA analysis, and (2) computational expense of each simulation run (high-fidelity code/s that employs the physic/s of interest). RAVEN implements both approaches by developing reduced order models (ROMs). ROMs are mathematical models of fast evaluation (approximately milliseconds) that can be trained by a given set of already performed samples of the system, using a blend of regression and interpolation techniques. The ROMs can answer both the challenges previously mentioned. ROMs can be built to seek for the minimum set of samples that allow determining the probability associated to a specific outcome of the system (i.e., in adaptive sampling methodologies). In addition, they can be built to represent the original physical model (high-fidelity code), replacing the simulation code itself and therefore providing a very fast tool to evaluate the system response. It needs to be noticed that, obviously, ROMs approximate the simulation code, and therefore the answer they provide is always affected by an error.

The milestone reported in June 2014 [9] described the infrastructure of the RAVEN code, presenting all the capabilities available at that time. The initial adaptive sampling strategy (limit surface [LS] search), in RAVEN, was already available and explained at the time. In the following year of development, the RAVEN adaptive sampling method was improved with more sophisticated convergence acceleration techniques. This milestone report illustrates these acceleration methodologies.

This report is structured as follows:

- Section 2 gives a brief overview of the RISMC approach
- Section 3 gives an overview of the RAVEN code with its main components
- Section 4 introduces the concept of LS, the search algorithm, how such methodology is implemented, and the acceleration methods subject of this report
- Section 5 presents a series of test cases to prove the validity of acceleration schemes implemented compared to the previous adaptive sampling approach and classical sampling methodologies
- Section 6 highlights the possible future development paths
- Section 7 presents conclusions.

2. RISK-INFORMED SAFETY MARGIN CHARACTERIZATION APPROACH

The RISMC pathway develops and delivers approaches to manage safety margins. [7] This important information supports NPP owner/operator decision-making associated with near- and long-term operation. The RISMC approach can optimize plant safety and performance by incorporating a novel interaction between probabilistic risk simulation and mechanistic codes for plant-level physics. The new approach allows the risk evaluation tool (e.g., RAVEN) to serve as a "scenario generator" that feeds information to the mechanistic codes. The new approach fits with the intrinsic goals of the RISMC pathway to:

1. Develop and demonstrate a risk-assessment method coupled to safety margin quantification. Decision-makers can use such methodology as part of their margin management strategies.
2. Create an advanced RISMC toolkit. This RISMC toolkit would enable a more accurate representation of an NPP safety margin and its associated influence on operations and economics.

When evaluating the safety margin, not only does the frequency of an event (e.g., core damage) need assessed, but also the system "probabilistic distance" to safety-related events and how it is possible to increase this distance through proper application of risk-informed margin management. In general terms, a "margin" is usually characterized either in a deterministic or probabilistic flavor. In a deterministic fashion, it is defined by the ratio (or, alternatively, the difference) of system capacity (i.e., strength) over (minus) the load to which the system is exposed. In a probabilistic fashion, it is defined by the probability that the load exceeds the capacity (in percentage or absolute value). A probabilistic safety margin is generated by the application of the above margin definition to a safety metric such as clad temperature (load) versus maximum clad temperature failure (capacity) in accident scenarios.

The RISMC pathway uses the probabilistic margin approach to quantify impacts to reliability and safety. As part of the quantification, both probabilistic (via risk simulation) and mechanistic (via physics models) approaches are used, as represented in Figure 1. Safety margin and uncertainty quantification rely on plant physics (e.g., thermal-hydraulics and reactor kinetics) coupled with probabilistic risk simulation. The coupling takes place through the interchange of physical parameters (e.g., pressures and temperatures) and operational or accident scenarios.

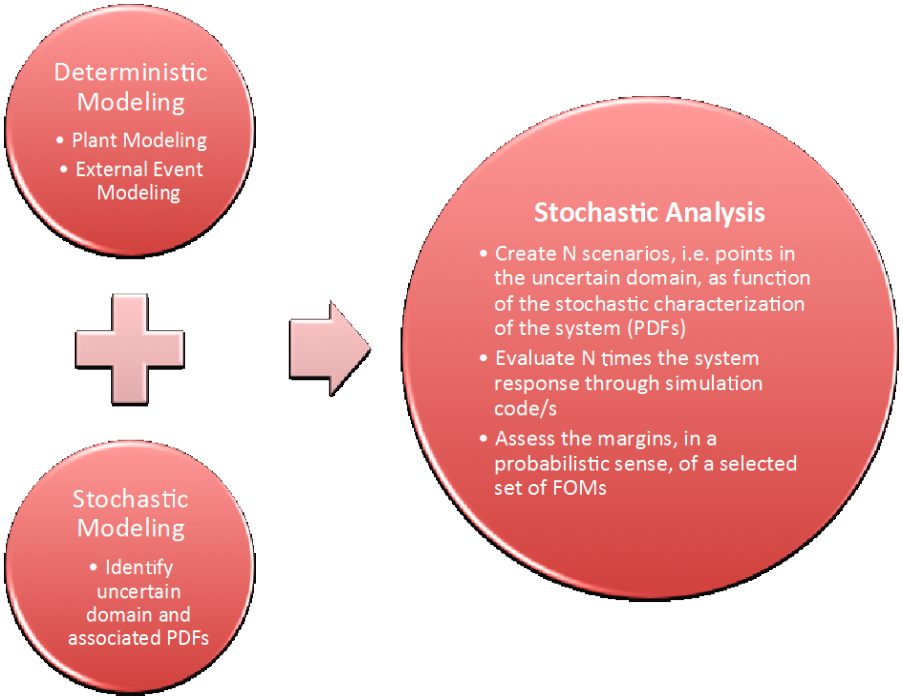

Figure 1. Schematic of risk-informed margin management approach.

As already mentioned, the RISMC approach heavily relies on multi-physics systems simulator codes (e.g., RELAP-7 [10]) coupled with stochastic analysis tools (e.g., RAVEN [1-6]).

By using the RISMC approach, the PRA analysis is performed by (see Figure 2):

1. Associating a probability distribution function (PDF) to the set of parameters $\boldsymbol{S}$, which include timing of events, initial conditions, and model parameters (e.g., friction coefficient)
2. Performing stochastic sampling of the PDFs defined in Step 1 to generate a realization of $\boldsymbol{s} \in \boldsymbol{S}$
3. Simulating the system response for each realization $\boldsymbol{s}$, generated in Step 2
4. Repeating Steps 2 and 3 M times and evaluating user-defined stochastic parameters.

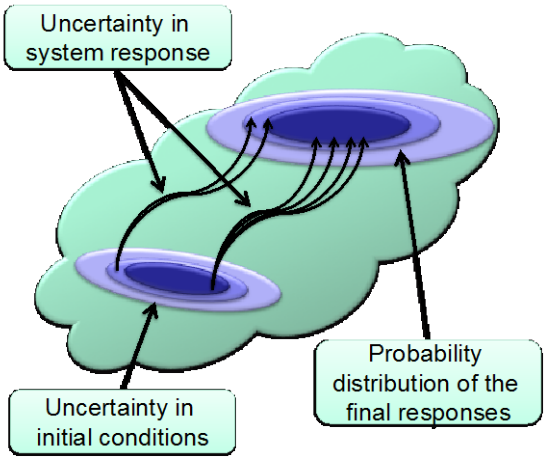

Figure 2. Stochastic dynamic system.

In this methodology environment, the RAVEN code represents the tool in charge for performing such analysis and, the smart sampling algorithms (e.g., LS search) represent essential methods to increase its effectiveness.

3. RAVEN FRAMEWORK

3.1 Introduction

As inferred from the initial introduction, the LS search algorithm and acceleration schemes, subjects of this report, were implemented and assessed within the probabilistic and uncertainty quantification framework, RAVEN. Hence, it is helpful to provide a brief overview of the code and its main capabilities and internal structure.

RAVEN was developed in a highly modular and pluggable way to enable easy integration of different programming languages (i.e., C++ and Python) and coupling with any system/physic code. Its main goal is to provide a tool to allow exploration of the uncertain domain, dispatching several different capabilities in an integrated environment.

3.2 Software Infrastructure Overview

The main idea behind the design of the RAVEN software package is the creation of a multi-purpose framework characterized by high flexibility with respect to the possible set of analysis that a user might request. To obtain this result, the code infrastructure must be capable of constructing the analysis/calculation flow at run-time, interpreting the user-defined instructions, and assembling the different analysis tasks following a user-specified scheme.

The need to achieve such flexibility, combined with reasonably fast development, pushed toward the programming language that is naturally suitable for this kind of approach: Python.

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

- ***Distribution:*** The probability of a specific system outcome is related to the probability of the set of input parameters and initial conditions that led to such outcome. Moreover, some sampling techniques (e.g., Monte-Carlo [MC]) explore the input space influenced by the probabilistic distribution associated to the input variables. Consequently, RAVEN possess a large library of PDFs.
- ***Sampler:*** A proper approach to sample the input space is fundamental for optimizing the computational time. In RAVEN, a "sampler" determines a unique perturbation strategy that is applied to the input space of a system. The association of uncertain variables and their corresponding probability distributions constitute the probabilistic input space on which the sampler operates.
- ***Model:*** A model is the representation of a physical system (e.g., NPP); it is therefore capable of predicting the evolution of a system given a coordinate set in the input space (i.e., the initial condition of the system phase space).
- ***ROM:*** The evaluation of the system response, as a function of the coordinates in the uncertain domain (also known as input space), is very computationally expensive, which makes brute-force approaches (e.g., MC methods) unpractical. ROMs are used to lower this cost by reducing the number of needed points and prioritizing the area of the uncertain domain that needs to be explored. They are a pure mathematical representation of the link between the input and output spaces for a particular system.

The list above is not comprehensive of all the RAVEN framework components, which also include visualization and storage infrastructure, statistical post-processors, and a data mining suite.

3.2.1 Distribution Entity

As already mentioned, the perturbation of the input space (initial conditions/parameters affected by uncertainties) needs to be performed to account for their probabilistic distributions. RAVEN provides, through an interface to the BOOST library, the following univariate (truncated and not) distributions:

- Bernoulli
- Binomial
- Exponential
- Logistic
- Lognormal
- Normal
- Poisson
- Triangular
- Uniform
- Weibull
- Gamma
- Beta
- Categorical.

The use of univariate distributions for sampling initial conditions is based on the assumption that the uncertain parameters are not correlated with each other. Quite often uncertain parameters are subject to correlations and thus the univariate approach is not applicable. This happens when a generic outcome depends on multiple variables or vice versa the outcome dependency description cannot be collapsed to a function of a single variable. RAVEN currently supports N-dimensional (N-D) PDFs both in the form of multivariate normal distribution and user-provided PDFs. The user can provide files containing the distribution values on either Cartesian or sparse grid. Depending on the grid structure used to provide the distribution values, RAVEN determines the interpolation algorithm used in the evaluation of the imported cumulative distribution function (CDF)/PDF:

- N-D spline [11] for Cartesian grids
- Inverse weight [12] for sparse grids.

Internally, RAVEN provides the needed N-D differentiation and integration algorithms to compute the PDF from the CDF and vice versa. This is needed to cover both cases where the user provides the PDF or CDF.

As already mentioned, the sampling methods use the distributions to perform probability-weighted perturbations. For example, in the MC approach, a random number $\in [\mathbf{0,1}]$ is generated (probability threshold) and the CDF, corresponding to that probability, is inverted to retrieve the parameter value usable in the simulation. The existence of the inverse for univariate distributions is guaranteed by the monotonicity of the CDF. For N-D distributions, this condition is not sufficient since the $\boldsymbol{CDF}(\boldsymbol{X}) \rightarrow [\mathbf{0}, \mathbf{1}], \boldsymbol{x} \in \boldsymbol{R^N}$ and therefore, it could not be a bijective function. From an application point of view, this means the inverse of an N-D CDF is not unique.

As an example, Figure 3 shows a multivariate normal distribution for a pipe failure as a function of the pressure and temperature. The plane identifies an iso-probability surface (in this case, a line) that

represents a probability threshold of 50% in this example. Hence, the inverse of this CDF is an infinite number of points.

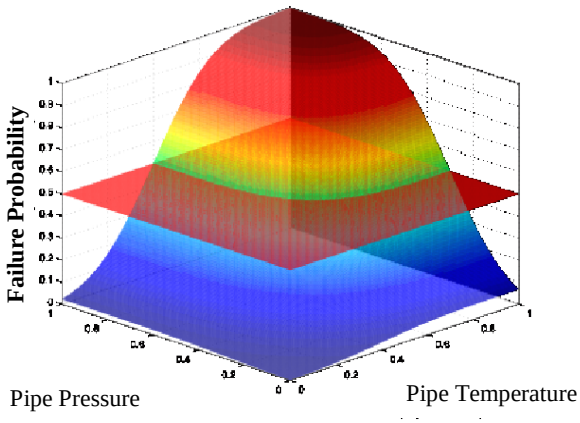

Figure 3. Example of two-dimensional multivariate probability distribution function.

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

3.2.2 Sampler

As already mentioned, the sampler is a key entity in the RAVEN framework to employ most of its capabilities of analysis. Indeed, it performs the driving of the specific sampling strategy and, hence, determines the effectiveness of the analysis, from both an accuracy and computational point of view. The samplers, that are available in RAVEN, are categorized in three main classes:

1. Forward
2. Dynamic event tree (DET)
3. Adaptive.

The following subsections briefly introduce the forward and DET samplers. As the adaptive samplers are the subject of this report, they are addressed separately in more detail. It is also worth mentioning that the adaptive samplers that are the subject of this report are focused on the search of the LS, while there is a parallel effort internally founded at Idaho National Laboratory aimed to construct an adaptive sampler for the full representation of the system response by polynomial interpolation. Unfortunately the only reference present at this time on this work is the RAVEN manual. [3]

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

In the RAVEN framework, several different forward samplers are available:

- MC
- Stratified (if equally spaced in probability ->LHS)
- Grid based
- Factorial designs:

- Full factorial
- Two-level fractional-factorial
- Plackett-Burman

- Response surface designs:
  - Box-Behnken
  - Central composite
- Stochastic collocation.

Since most of the forward sampling strategies previously listed are well known, they are not fully described in this report. More details regarding the MC, stratified, and grid sampling strategies are found in Ref. [9]; details regarding the factorial and response surface designs are found in Ref. [4]. In conclusion, detailed information about the stochastic collocation method is found in Ref. [13].

3.2.2.2 Dynamic Event Tree Sampler.

To clarify the idea behind the DET sampler currently available in RAVEN, a brief overview is needed.

In technological complex systems, such as NPPs, an accident scenario begins with an initiating event and then evolves over time through the interaction of deterministic and stochastic events. This mutual action leads to the production of infinitely many different outcomes. When for the same point in the input space the system might generate an infinite number of final status of the system, the system generates a continuous DET with infinite branches. At each time along the trajectory of the system in the phase space, the system might take a different path that is determined by a multivariate PDF. Since the continuous problem is almost untreatable, an approximate approach is needed to perform the PRA analysis. An approximation alternative is offered by the event tree approach or more recently by the DET approach.

In PRA analysis, in the conventional event tree [14] approaches, branches are used to differentiate among different statuses of the system and they do not have a temporal meaning (e.g., auxiliary generator working/not working). This approach lacks the capability to evaluate the impact of timing of the transition between different statuses of the system (in reality some treatment is possible but in a very costly fashion). To overcome these limitations, a "dynamic" approach is needed. The DET [14] technique brings several advantages, among which is the fact that it simulates probabilistic system evolution in a way that is consistent with its deterministic time evolution. This is done by taking the timing of events explicitly into account, leading to a more realistic and mechanistically consistent analysis of the possible evolution the system. This feature of the DET is very important, for example, when the complexity of the system leads to strong non-linear responses that characteristically evolve over time (the non-linear structure of the system strongly changes during the time evolution of the scenario). This result is obtained by "letting" the system code determine the pathway of an accident scenario within a probabilistic "environment."

From an application point of view, an N-D grid is built on the CDF space that is constructed by a tensor product of the CDF corresponding to each probabilistic variable characterizing the system. Those probabilistic variables represent coordinates of the phase space of the system that have a probability of changing from their initial value; the change probability is a function of the overall system status and time. When the system simulation starts, a complex system of controls (trigger system) monitor the system evolution and in particular the transition CDF for those probabilistic variables given the overall status of the system and time. When the CDF of one of those variables reaches the border of an N-D cell of the grid in the CDF space, a second simulation is started where the transition of the probabilistic variable has effectively taken place while the original simulation advance in time without any change in the variable. Each simulation carries along its own conditional probability. A more complete description of this methodology is found in Ref. [14].

Figure 4 shows a practical DET example. In this particular case, it is assumed that the probability failure of a pipe depends on the fluid pressure magnitude. Three probability thresholds are defined on the CDF. One simulation is spawned (0). As soon as the pressure of the fluid reaches a value corresponding to a 33% probability (CDF), a stop signal is sent and the framework starts two new simulations (branches). The branch in which the system evolved to the newer condition (pipe failed, red line) carries 33% of the probability, while the other the complementary amount. The same procedure is repeated at point 2.

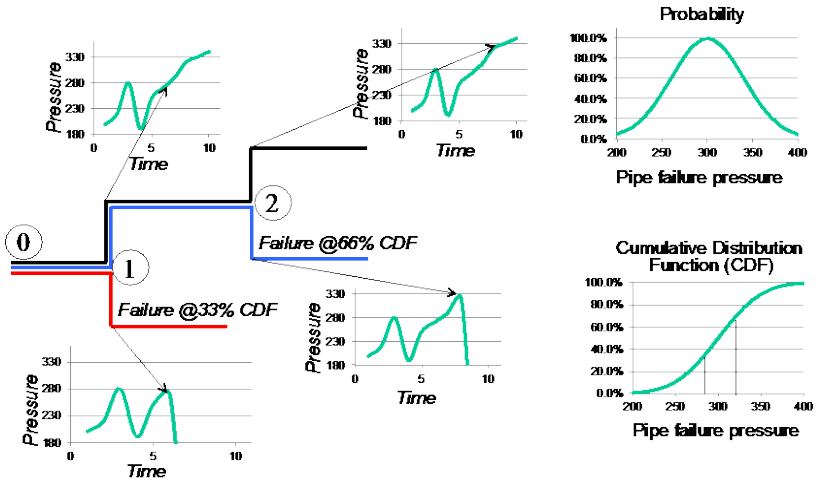

Figure 4. Example of dynamic event tree.

Generally, not all the input space can be explored using a DET approach. For instance, usually the parameters affected by aleatory uncertainty are sampled using a DET approach, while the ones characterized by epistemic uncertainty are sampled through forward sampling strategies. At the moment a hybrid approach (forward sampling of initial conditions followed by a DET strategy) is available.

As already mentioned, this strategy requires a tight interaction between the system code and sampling driver (i.e., RAVEN framework). In addition, the system code must have a control logic capability (i.e., trigger system). For these reasons, the application of this sampling approach to a generic code needs more effort when compared to the other samplers available in RAVEN. Currently, the DET is fully available for the thermal-hydraulic codes RELAP-7 and RELAP5-3D. [15]

3.2.2.3 Models. The model entity, in the RAVEN environment, represents a "connection pipeline" between the input and output spaces. The RAVEN framework does not own any physical model (i.e., it does not possess the equations needed to simulate any physical system), but implements application program interfaces (APIs) by which these models are supplied by the users. The RAVEN framework provides APIs for three different model categories:

- Codes
- Externals
- ROMs.

The code model represents the communication pipe between the RAVEN and any external software.

Currently, RAVEN has implemented APIs for RELAP5-3D, RELAP-7, any Multi-Physics Object-Oriented Simulation Environment-based [16] application, and the PHISICS code [17].

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

The data exchange between RAVEN and the system code is performed by direct software interface or files. If the system code is parallelized, data exchange by files is generally the way to follow since it is much more optimized in large clusters.

Since the ROMs are key tools for the LS search acceleration schemes, they are addressed in Section 4.

3.2.3 Simulation Environment

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

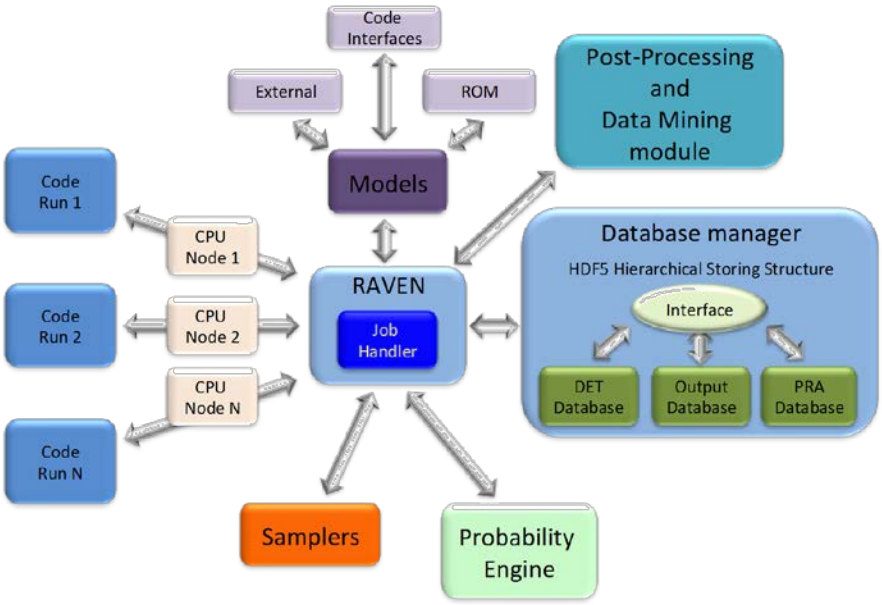

Figure 5. RAVEN schematic module interaction.

From a user's standpoint, RAVEN is perceived as a pool of tools and data. Any action in which the tools are applied to the data is considered a 'step' in the RAVEN environment. Since this report is focused on LS algorithms, that require sampling of the uncertain domain, only the "step" that is designed for this task (multiRun) is mentioned.

The "multiRun" step is designed to manage several runs (sampling) to explore a single model. At the beginning of each subsequential run, the sampler provides the new values of the variables to be modified. The code API places those values properly in the code input file, generates the run command, and asks the job handler (RAVEN's module) to queue the corresponding run. The job handler manages the parallel execution of as many runs as possible within a user-prescribed amount of computational resources. It also informs the multiRun step when a new set of output files, generated by one of the code runs, are ready to be processed. The multiRun step passes the new output files to the code API that collects the data in the RAVEN internal format. At the end, the sampler is queried to assess if the sequence of runs is ended, if not, the multiRun step controller asks for a new set of values from the sampler and the sequence is restarted.

The job handler is currently capable to run different instances of the code in parallel and can also handle codes that are internally multithreaded or using any form of message passing interface parallel implementation.

RAVEN is also capable of plotting the simulation outcomes while the set of sampling is performed and storing the data for later recovery.

4. LIMIT SURFACE SEARCH

As already mentioned, the key subject of this report is the development of convergence acceleration schemes for the LS search algorithm. LS search is one of the adaptive (or smart) sampling strategies developed within the RAVEN framework.

As briefly mentioned, the motivation of adaptive sampling strategies is that physic simulations are often computationally expensive, time-consuming, and with a large number of uncertain parameters. Thus, exploring the space of all possible simulation outcomes is almost infeasible using finite computing resources. During simulation-based PRA analysis, it is important to discover the relationship between a potentially large number of uncertain parameters and the response of a simulation using as few simulation trials as possible.

This is a typical context where “goal” oriented sampling could be beneficial. Among the different types of goal-oriented sampling, RAVEN uses a schema where few observations, obtained from the model run, are used to build a simpler and faster evaluable mathematical representation of the model (ROM). The ROM is then used to predict where further exploration of the input space could be most informative. This information is used to select new locations in the input space for which a code run is executed (see Figure 6). The new observations are used to update the ROM and this process iterates until, within a certain metric, it is converged.

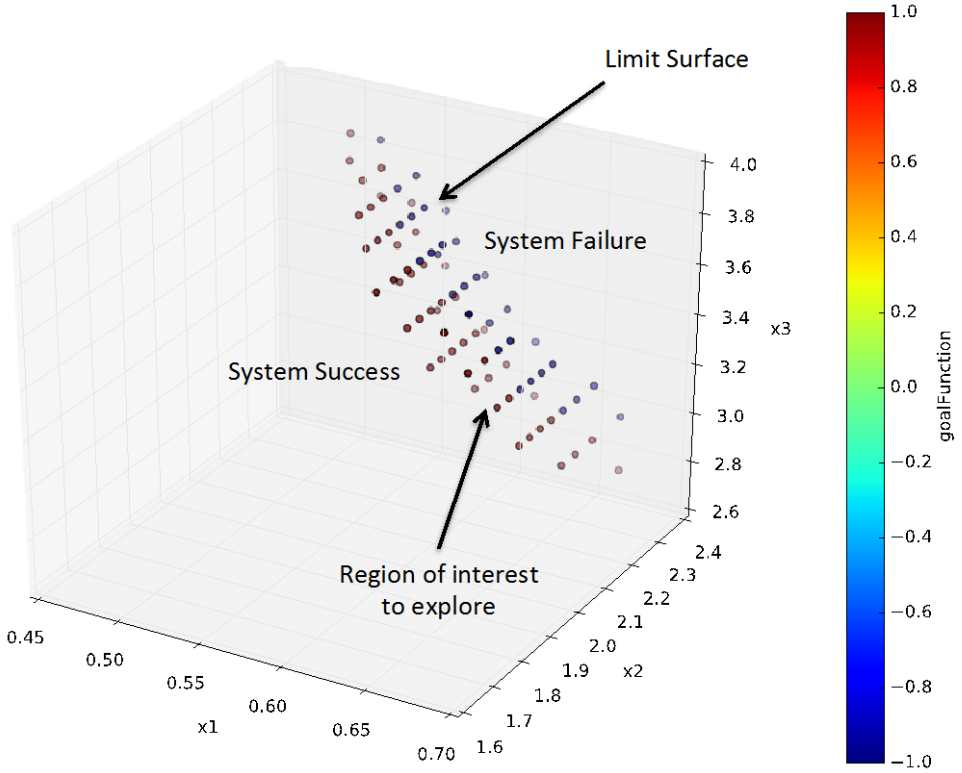

Figure 6. Example of limit surface schematic.

To summarize, in the case of the LS search, a ROM is used to determine which location in the input space further observations is most informative, to establish the location of the LS, then code runs are executed on those locations and the ROM updated. The process continues until the location of the LS is established within a certain tolerance.

4.1 Limit Surface Concept and Properties

As already mentioned, this report describes the acceleration schemes implemented for the research of the LS. [18] To properly explain these acceleration algorithms, it is necessary to analyze the concept of the LSs, firstly, from a mathematical and, secondly, from a practical point of view.

Consider a dynamic system that is represented in the phase space by:

$$\bar{\boldsymbol{y}} = \boldsymbol{H}(\bar{\boldsymbol{x}}, \boldsymbol{t}, \bar{\boldsymbol{p}}) \tag{1}$$

where

$\bar{\boldsymbol{y}}$ = coordinate of the system in the phase space

$\bar{\boldsymbol{x}}, \boldsymbol{t}, \bar{\boldsymbol{p}}$ = independent variables that are separated, respectively, in spatial, temporal, and parameters' independent variables (distinction between $(\bar{\boldsymbol{x}}, \boldsymbol{t}, \bar{\boldsymbol{p}})$ is purely based on engineering considerations).

Now it is possible to introduce the concept of "goal" function, $\boldsymbol{C}$. $\boldsymbol{C}$ is a binary function that, based on the response of the system, can assume the value $\boldsymbol{0}$ (false) to indicate that the system is properly available (e.g., system success) and $\boldsymbol{1}$ (true) to testify that the system is not available (e.g., failure of the system):

$$\boldsymbol{C} = \boldsymbol{C}(\bar{\boldsymbol{y}}, \bar{\boldsymbol{x}}, \boldsymbol{t}, \bar{\boldsymbol{p}}) = \boldsymbol{C}(\boldsymbol{H}(\bar{\boldsymbol{x}}, \boldsymbol{t}, \bar{\boldsymbol{p}}), \bar{\boldsymbol{x}}, \boldsymbol{t}, \bar{\boldsymbol{p}}) = \boldsymbol{C}(\bar{\boldsymbol{x}}, \boldsymbol{t}, \bar{\boldsymbol{p}}) \tag{2}$$

To simplify the dissertation in this report and without loss of generality, assume that $\boldsymbol{C}$ does not depend on time (e.g., $\boldsymbol{C} \leftarrow \int_{\boldsymbol{t_0}}^{\boldsymbol{t_{end}}} \boldsymbol{dtC}\,(\bar{\boldsymbol{x}}, \boldsymbol{t}, \bar{\boldsymbol{p}})$):

$$\boldsymbol{C} = \boldsymbol{C}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{p}}) \tag{3}$$

To simplify the mathematical description of the LS concept, it is possible to hypothesize that the equation describing the PDF time evolution of the system in the phase space is of type Gauss-Codazzi (in its Liouville's derivation), which allows ensuring that all the stochastic phenomena in the system are representable as PDFs in the uncertain domain. This allows combining the parametric space with the initial condition space:

$$\begin{aligned} &(\bar{\boldsymbol{x}}) \leftarrow (\bar{\boldsymbol{x}}, \bar{\boldsymbol{p}}) \\ &\boldsymbol{C}(\bar{\boldsymbol{x}}) \leftarrow \boldsymbol{C}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{p}}) \end{aligned} \tag{4}$$

This assumption is rarely violated, for example for those systems that present an intrinsic stochastic behavior (e.g., the dynamic of a particle of dust in the air where it continuously and randomly interacts with the molecules of air that "move" with different velocities and in different and random directions). In most of the cases of interest in the safety analysis, the above-mentioned assumption is correct. A heuristic approach to the characterization of system stochastic behaviors is reported in Ref. [6].

Under the above simplifications, it is possible to identify the region of the input space ($\boldsymbol{V}$) leading to a specific outcome of the "goal" function. In particular, it can be defined, for example, the failure region $\boldsymbol{V_F}$ as the region of the input space where $\boldsymbol{C = 1}$:

$$\boldsymbol{V_F} = \{\forall \overline{\boldsymbol{x}} | \boldsymbol{C}(\overline{\boldsymbol{x}}) = \boldsymbol{1}\} \tag{5}$$

The definition of the complementary of the failure region is obviously:

$$\boldsymbol{V_F^c} = \{\forall \overline{\boldsymbol{x}} | \boldsymbol{C}(\overline{\boldsymbol{x}}) = \boldsymbol{0}\} \tag{6}$$

Its boundary is the named LS:

$$\boldsymbol{L_s} = \boldsymbol{\partial V_F} = \boldsymbol{\partial}\{\forall \overline{\boldsymbol{x}} | \boldsymbol{C}(\overline{\boldsymbol{x}}) = \boldsymbol{1}\} \tag{7}$$

The identification of the LS location is necessary to identify boundary regions for which the system under consideration will or will not exceed certain FOMs (e.g., operative margins).

The LS location is extremely important for design optimization and issue mitigation and, in addition, its informative content can be used to analyze the system to characterize its behavior from a stochastic point of view. Consider $\overline{\boldsymbol{x}} \in \boldsymbol{V}$ and $\overline{\boldsymbol{x}} \sim \overline{\boldsymbol{X}}$, where $\overline{\boldsymbol{x}}$ is the random variate realization of the stochastic variable $\overline{\boldsymbol{X}}$.

If $\boldsymbol{pdf}_{\overline{\boldsymbol{X}}}(\overline{\boldsymbol{x}})$ is the probability density function of $\overline{\boldsymbol{X}}$, the failure probability of the system ($\boldsymbol{P_F}$) is:

$$\boldsymbol{P_F} = \int_{\boldsymbol{V}} \boldsymbol{d\overline{x} C}(\overline{\boldsymbol{x}}) \boldsymbol{pdf}_{\overline{\boldsymbol{X}}}(\overline{\boldsymbol{x}}) = \int_{\boldsymbol{V_F + V_F^c}} \boldsymbol{d\overline{x} C}(\overline{\boldsymbol{x}}) \boldsymbol{pdf}_{\overline{\boldsymbol{X}}}(\overline{\boldsymbol{x}}) \tag{8}$$

And, based on the definition given in *Equations (4)* and *(5)*:

$$\boldsymbol{P_F} = \int_{\boldsymbol{V_F}} \boldsymbol{d\overline{x}\, pdf}_{\overline{\boldsymbol{X}}}(\overline{\boldsymbol{x}}) \tag{9}$$

Equations (8) and *(9)* are summarized by stating that the system failure probability is equivalent to the probability of the system being in the uncertain subdomain (region of the input space) that leads to a failure pattern. This probability is equal to the probability-weighted hyper-volume that is surrounded by the LS.

It is beneficial for the reader to assess the LS concept through an example related to the safety of an NPP.

As an example, consider a station black-out (SBO) scenario in an NPP. Suppose that the only uncertain parameters are:

- $\boldsymbol{t_F}$: Temperature that would determine the failure of the fuel cladding
- $\boldsymbol{rt_{DGs}}$: Recovery time of the diesel generators (DGs) that can guarantee, through the emergency core cooling system (ECCS), the removal of the decay heat.

And, the corresponding CDF (uniform) is:

$$\boldsymbol{t_F \sim pdf_{T_F}(T_F)} = \begin{cases} \boldsymbol{= 0 \; if \; t_F < t_{F_{min}}} \\ \boldsymbol{= \dfrac{1}{(t_{F_{max}} - t_{F_{min}}) = \Delta t_F}} \\ \boldsymbol{= 0 \; if \; t_F > t_{F_{max}}} \end{cases} \tag{10}$$

$$\boldsymbol{rt_{DGs} \sim pdf_{RT_{DGs}}(rt_{DGs})} = \begin{cases} \boldsymbol{= 0 \; if \; rt_{DGs} < rt_{DGs_{min}}} \\ \boldsymbol{= \dfrac{1}{(rt_{DGs_{max}} - rt_{DGs_{min}}) = \Delta rt_{DGs}}} \\ \boldsymbol{= 0 \; if \; rt_{DGs} > rt_{DGs_{max}}} \end{cases} \tag{11}$$

For simplicity, assume that the clad temperature is a quadratic function of the DG recovery time in an SBO scenario:

$$\boldsymbol{t = t_0 + \alpha \, rt_{DGs}^2} \tag{12}$$

and that the $\boldsymbol{t_{F_{min}} > t_0 + \alpha \, rt_{DGs_{min}}^2}$, $\boldsymbol{t_{F_{max}} < t_0 + \alpha \, rt_{DGs_{max}}^2}$.

The LS, failure region, and active part of the failure region (failure region with non-zero probability) are illustrated in Figure 7 (in agreement with the above assumptions).

In this case, the transition/failure probability is evaluated as follows:

$$\boldsymbol{P_F = \int_{V_F} pdf_{\overline{X}}(\overline{x}) d\overline{x} \; = \; \int_0^{+\infty} pdf_{T_F}(T_F) dt_F \int_{\sqrt{\frac{t_F - t_0}{\alpha}}}^{+\infty} pdf_{RT_{DGs}}(rt_{DGs}) \, drt_{DGs}}$$

$$\boldsymbol{P_F = \int_{t_{F_{min}}}^{t_{F_{max}}} \frac{1}{t_{F_{max}} - t_{F_{min}}} dt_F \int_{\sqrt{\frac{t_F - t_0}{\alpha}}}^{rt_{DGs_{max}}} \frac{1}{rt_{DGs_{max}} - rt_{DGs_{min}}} drt_{DGs}} \tag{13}$$

$$\boldsymbol{P_F = \frac{rt_{DGs_{max}}}{\Delta rt_{DGs}} + \frac{2\alpha}{3(\Delta rt_{DGs} \Delta t_F)} \left(\sqrt[3/2]{\frac{t_{F_{min}} - t_0}{\alpha}} - \sqrt[3/2]{\frac{t_{F_{max}} - t_0}{\alpha}} \right)}$$

This simple example is useful to understand how the LS is defined in a practical example (that is analyzed numerically in the results section) and how the hyper-volume needs weighted with respect to the probability in the uncertain domain. An example of the LS computed by RAVEN, using RELAP-7, is shown in Figure 7.

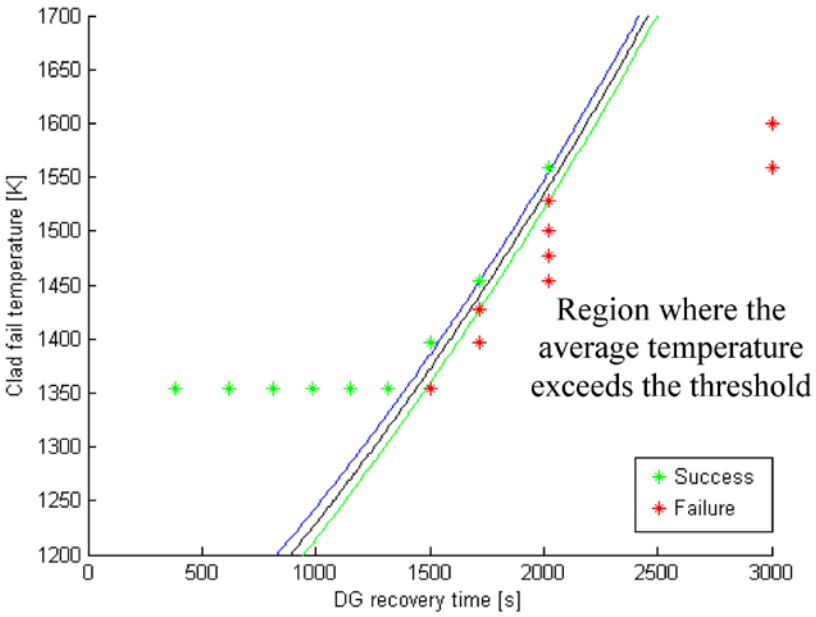

Figure 7. Example of limit surface.

4.2 Reduced Order Models

As briefly and previously mentioned, a ROM, also called a surrogate model, is a mathematical representation of a system, used to predict a selected FOM of a physical system.

The "training" is a process that, sampling the physical model represented by the high-fidelity simulator (e.g., RELAP-7, RELAP5-3D, and PHISICS), is aimed to improve the prediction capability (capability to predict the status of the system given a realization of the uncertain domain) of the ROM. Two characteristics of these models are generally assumed (even if exceptions are possible):

1. The higher the number of realizations in the training sets, the higher the accuracy of the prediction performed by the ROM is. This statement is true for most of the cases although some ROMs might be subject to the over-fitting issues. The over-fitting phenomenon is not analyzed in this report, since its occurrence highly depends on the algorithm type, and, hence, the problem needs to be analyzed for all the large number of ROM types currently available in RAVEN. Currently, it is advisable that the user, that chooses a particular ROM construction algorithm, consults the relative literature. As reported in Section 6, in the near future, the RAVEN team is planning to address this problem, identifying and implementing algorithms for the "automatic" calibration of all the ROMs' input parameters.
2. The smaller the size of the input domain with respect to the variability of the system response, the more likely the ROM is able to represent the system response space.

To provide a very simple idea of a ROM, assume that the final response space of a physical system is governed by the transfer function $\boldsymbol{H}(\overline{\boldsymbol{x}})$, that, from a practical point of view, represents the outcome of the system, based on the initial conditions $\overline{\boldsymbol{x}}$. Now, sample the domain of variability of the initial conditions $\overline{\boldsymbol{x}}$ to create a set of $\boldsymbol{N}$ realizations of the input and response space $\left(\overline{\boldsymbol{x}}_{\boldsymbol{i}}, \boldsymbol{H}(\overline{\boldsymbol{x}}_{\boldsymbol{i}})\right), \boldsymbol{i} = \boldsymbol{1}, \boldsymbol{N}$, named "training" set. Based on the data set generated, it is possible to construct a mathematical representation $(\boldsymbol{G}(\overline{\boldsymbol{x}}) : \overline{\boldsymbol{x}}_{\boldsymbol{i}})$ of the real system $\boldsymbol{H}(\overline{\boldsymbol{x}})$, which will approximate its response (see Figure 8):

$$\boldsymbol{G}(\overline{\boldsymbol{x}}) : \overline{\boldsymbol{x}}_{\boldsymbol{i}} \rightarrow \boldsymbol{G}(\overline{\boldsymbol{x}}_{\boldsymbol{i}}) \cong \boldsymbol{H}(\overline{\boldsymbol{x}}_{\boldsymbol{i}}) \tag{14}$$

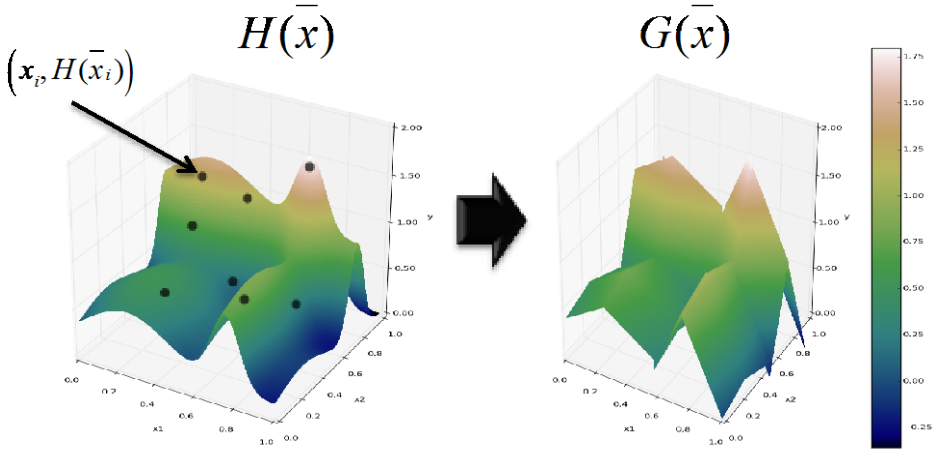

Figure 8. Example of reduced order model representation of physical system (regression).

The ROMs reported above are generally named "regressors," among which all the most common data fitting algorithms are found (e.g., least-square for construction of linear models).

An important class of ROMs for the work presented hereafter is the one containing the so-called "classifiers." A classifier is a ROM that is capable of representing the system behavior from a binary point of view (e.g., event happened/not happened or failure/success). It is a model (set of equations) that identifies to which category an object belongs in the feature (input) space. Referring to the example that brought to *Equation (14)*, a classifier is represented as follows (see Figure 9):

$$\boldsymbol{G}(\bar{\boldsymbol{x}}): \bar{\boldsymbol{x}}_{\boldsymbol{i}} \rightarrow \boldsymbol{G}(\bar{\boldsymbol{x}}_{\boldsymbol{i}}) \cong \boldsymbol{C}(\boldsymbol{H}(\bar{\boldsymbol{x}}_{\boldsymbol{i}})) \tag{15}$$

The function $\boldsymbol{C}(\boldsymbol{H}(\bar{\boldsymbol{x}}_{\boldsymbol{i}}) = \bar{\boldsymbol{y}})$ is the so-called "goal" function that is able to recast the response of the system $\boldsymbol{H}(\bar{\boldsymbol{x}})$ into a binary form (e.g., failure/success). As an example, referring to Figure 9, the "goal" function would be:

$$\boldsymbol{C}(\bar{\boldsymbol{y}}) = \begin{cases} \mathbf{1}\ \boldsymbol{if}\ \bar{\boldsymbol{y}} > \mathbf{1.0} \\ \mathbf{0}\ \boldsymbol{if}\ \bar{\boldsymbol{y}} \leq \mathbf{1.0} \end{cases} \tag{16}$$

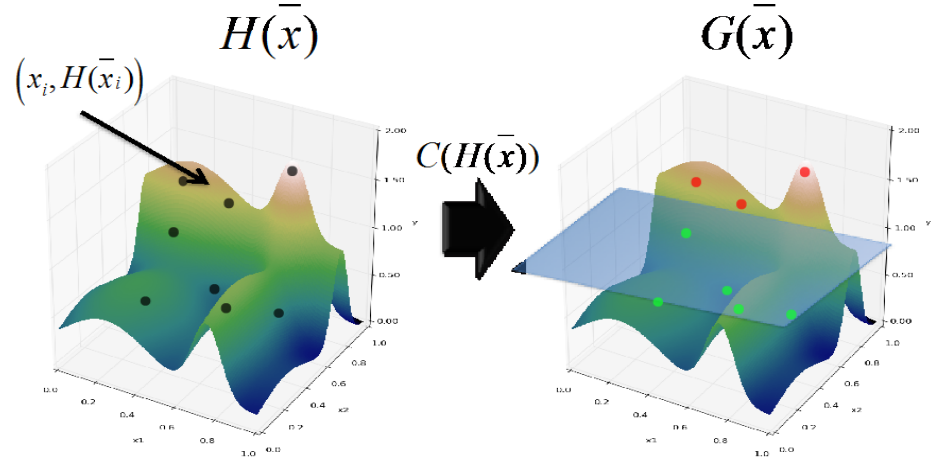

Figure 9. Example of reduced order model representation of physical system (classifier).

Hence, the ROM of type classifier $\boldsymbol{G}(\bar{\boldsymbol{x}})$ will operate in the space transformed through the "goal" function $\boldsymbol{C}(\bar{\boldsymbol{y}})$.

The classifiers and regressors currently available in RAVEN are organized in two main classes:

1. Model-based algorithms
2. Data-based algorithms.

In the first class, the created ROM aims to approximate the response of the system as a function of the input parameters. These algorithms construct a functional representation of the system. In RAVEN there are several different types of model-based algorithms, such as support vector machines (SVMs), Kriging-based interpolators, discriminant-based models, and polynomial chaos.

On the other side, data-based algorithms do not build a response-function-based ROM but classify or predict the response of the system from the neighborhood graph constructed from the training data, without any dependencies on a particular prediction model.

These algorithms directly build a neighborhood structure as the ROM (e.g., a relaxed Gabriel graph) on the initial training data. In RAVEN, there are several different types of data-based algorithms, such as nearest neighbors and decision trees.

4.3 Limit Surface Search Algorithm

Determination of the LS location is extremely challenging, depending on the particular physics/phenomena that are investigated. To identify the real location of the LS, evaluation of system responses is needed, through the high-fidelity code (e.g., RELAP-7, and RELAP5-3D), in the full domain of uncertainty (infinite number of combinations of uncertainties represented by the respective PDFs). Obviously, this is not a feasible approach, and a reasonable approximation is to locate the LS on a Cartesian N-D grid, in the uncertain domain.

In reality, the location of the LS is not exactly determined but rather bounded. The algorithm determines the set of grid nodes between which the transition 0/1 of the "goal" function happens. This set is also classified with respect to the value of the "goal" function. With reference to Figure 10, for example, green is used for grid nodes with a "goal" function that equals 0 and red when the "goal" function equals 1.

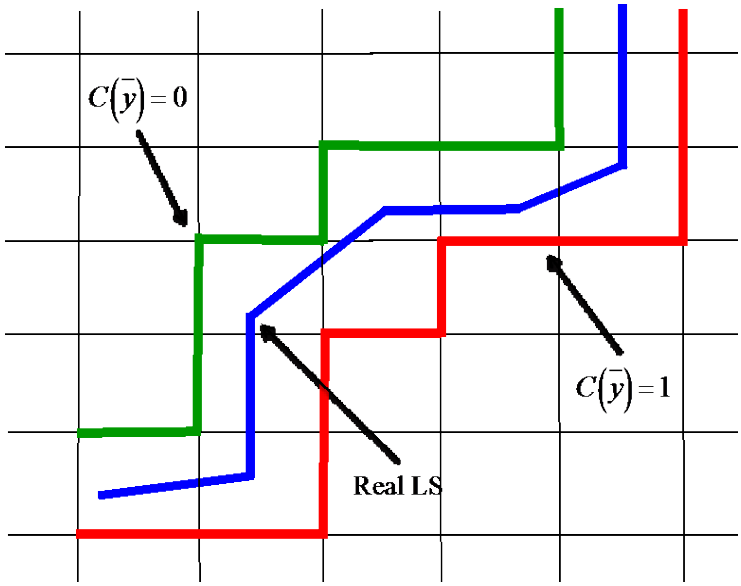

Figure 10. Example of limit surface search evaluation grid.

Each evaluation of the "goal" function in one of the grid nodes implies the evaluation of the high-fidelity code (e.g., RELAP-7) for the corresponding set of entry in the uncertain space. As already mentioned, evaluation of the high-fidelity code is computationally expensive and, in order to identify the LS, RAVEN should appraise each point in the N-D grid covering the uncertainty space. Discretization depends on the accuracy requested by the user. In most cases, this approach is not feasible and, consequentially, the process needs accelerated by using "predicting" methods that, in RAVEN, are represented by the employment of supervised learning algorithms.

What RAVEN implements is, in reality, what is commonly referred to as an active learning process that ultimately results in training of a ROM of type classifier capable of predicting the outcome of the "goal" function for any given point of the uncertain space.

In an active learning process, a supervised learning algorithm is combined with criteria to choose the next node in the N-D grid that needs explored, using the high-fidelity physical model. This process is repeated until, under a particular metric, the prediction capabilities of the supervised learning algorithm do not improve by further increasing the training set.

In more detail, the iterative scheme could be summarized through the following steps:

1. A limited number of points in the uncertain space $\{\overline{\boldsymbol{x}}_{\boldsymbol{k}}\}$ are selected via one of the forward sampling strategies (e.g., stratified or MC).
2. The high-fidelity code is used to compute the status of the system for the set of points in the input set: $\{\overline{\boldsymbol{x}}(\boldsymbol{t})\}_{\boldsymbol{k}} = \boldsymbol{H}(\{\overline{\boldsymbol{x}}\}_{\boldsymbol{k}}, \boldsymbol{t})$.
3. The "goal" function is evaluated at the phase space coordinate of the system: $\{\boldsymbol{c}\}_{\boldsymbol{k}} = \boldsymbol{C}(\{\overline{\boldsymbol{x}}(\boldsymbol{t})\}_{\boldsymbol{k}})$.
4. The set of pairs $\{(\overline{\boldsymbol{x}}, \boldsymbol{c})_{\boldsymbol{k}}\}$ are used to train a ROM of type classifier, $\boldsymbol{G}(\{\overline{\boldsymbol{x}}\}_{\boldsymbol{k}})$.
5. The ROM classifier is used to predict the values of the "goal" function for all the $\boldsymbol{N}$ nodes of the N-D grid in the domain space:

$$\boldsymbol{G}\left(\{\overline{\boldsymbol{x}}\}_{\boldsymbol{j}}\right) \approx \{\boldsymbol{c}\}_{\boldsymbol{j}}, \boldsymbol{j} = \boldsymbol{1}, \dots, \boldsymbol{N} \tag{17}$$

6. The values of the "goal" function are used to determine the LS location based on the change of values of $\{\boldsymbol{c}\}_{\boldsymbol{j}}$:

$$\{\boldsymbol{c}\}_{\boldsymbol{j}} \rightarrow \boldsymbol{\partial V_F} \tag{18}$$

7. A new point is chosen to increase the training set and a new pair is generated.
8. The procedure is repeated starting from Step 3 until convergence is achieved. The convergence is achieved when there are no changes in the location of the LS after a certain number of consecutive iterations.

The iteration scheme is graphically shown in Figure 11.

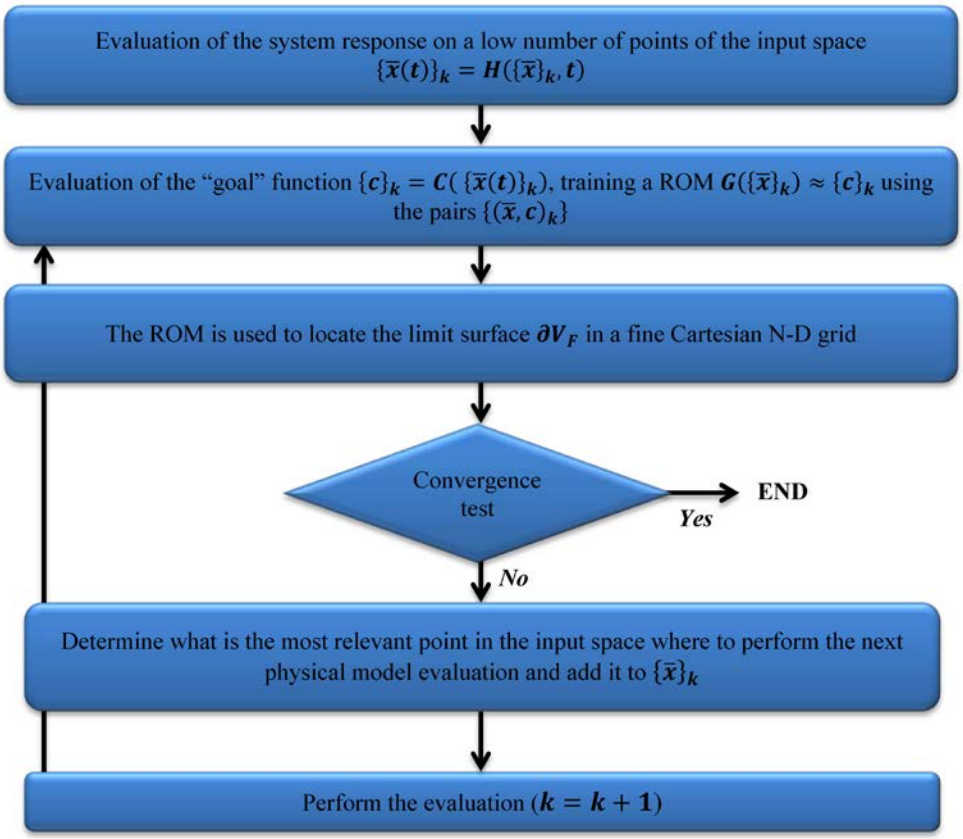

Figure 11. Limit surface search conceptual scheme.

It is important to notice that there is an additional requirement regarding the LS search algorithm. It is required that the LS location stays constant for a certain number (user-defined) of consecutive iterations. The reason for this choice is determine by the attempt to mitigate the effect of the build of non-linear bias in the searching pattern. Indeed, the searching algorithm might focus too much on a certain region of the LS while putting too few points in other zones and completely hiding undiscovered topological features of the LS.

Regarding the strategy to choose the nodes on the N-D grid that needs evaluated in the iterative process for the LS identification, currently RAVEN employs a metric based on the distance between the predicted LS and the evaluations already performed. The points on the LS are ranked based on the distance from the closest training point already explored (the larger is the distance the higher is the score for the candidate point), and based on its persistence (the larger is the number of time the prediction of the "goal" function for that point have changed the higher is the score).

Since this approach creates a queue of ranked candidates, it could be used also in the parallel implementation of the algorithm. When several training points are run in parallel, it is possible that the evaluation of one additional point does not alter dramatically the location of the LS. Consequently, it is possible that the candidate with the highest score is already being submitted for evaluation and possibly the simulation is not yet completed. In this case, to avoid submitting the same evaluation point twice, RAVEN will search among all the ranked candidates (in descending order) for the one that was not submitted for evaluation. Even if it is extremely unlikely that all the candidates were submitted, in this remote event, RAVEN will choose the next point employing an MC strategy.

4.4 Acceleration Schemes

This subsection addresses the acceleration schemes that were implemented. All of these algorithms are based on the devolvement of some propaedeutic capabilities, within the RAVEN framework, that are addressed in Subsection **Error! Reference source not found.**.

4.4.1 Propaedeutic Development

As already mentioned, the methodologies addressed in the following subsections are based on the development of some additional capabilities in the RAVEN code. Three main developments were crucial for development of the acceleration schemes:

- Grid entity
- Limit surface post-processor
- Limit surface integration post-processor.

The following subsections briefly explain these developments.

4.4.1.1 Grid Entity. Implementation of a grid entity, within the RAVEN framework, was very important to develop the acceleration algorithms later addressed in this report. In the past, RAVEN did not have an object that was fully dedicated in handling N-D Cartesian grids, but, every component of the code in need of such a structure was internally constructing its own. This approach created a certain level of redundancy and made the integration/interaction of multiple grid-based objects complex (e.g., LS search, factorial, and LS post-processor). For these reasons, development of a single object, capable of handling all the common needs regarding grid handling, was pursued and accomplished. The new entity can handle Cartesian and sparse grid.

4.4.1.2 Limit Surface Post-processor. Another important development is represented by implementation of a post-processor for computing the LS. This post-processor can be used to compute the LS, based on already-generated data. For example, it is generally used to generate the LS at the end of a non-adaptive sampling strategy (e.g., MC or factorial).

In addition, to remove redundancy, it is also directly used by the LS search sampling strategy.

4.4.1.3 Limit Surface Integration Post-processor. As already mentioned, the hyper-volume, which is bounded by the LS, is a measure of the probability of the event that is modeled through the "goal" function. For this reason, a post-processor was developed to compute the weighted (or not) integral of the LS.

The computation of the LS integral is currently performed employing an MC integration scheme.

4.4.2 Acceleration through Multi-grid Approach

The location of the LS, being a numerical iterative process, can be known given a certain tolerance. As already mentioned, the LS search is done by constructing an evaluation grid, on which the acceleration ROM is inquired. The tolerance of the iterative determines how the evaluation grid is discretized. Before addressing the new acceleration scheme, it is important to introduce some concepts on the employed numerical process.

Assume that each of $\boldsymbol{D}$ dimensions of the uncertain domain is discretized with the same number of equally-spaced nodes $\boldsymbol{N}$ (see Figure 12), with discretization size indicated by $\boldsymbol{h_i}$. Hence, the Cartesian grid contains $\boldsymbol{N^D}$ individual nodes, indexed through the multi-index vector $\bar{\boldsymbol{j}} = (\boldsymbol{j_{i=1\to D}}), \boldsymbol{j_i} \leq \boldsymbol{N} \forall \boldsymbol{i}$. Introducing the vectors $\bar{\boldsymbol{I}} = (\boldsymbol{1}, \dots, \boldsymbol{1})$ and $\bar{\boldsymbol{N}} = (\boldsymbol{N}, \dots, \boldsymbol{N})$, the "goal" function is expressed on this N-D grid as:

$$\boldsymbol{C}(\bar{\boldsymbol{x}}) = \sum_{\bar{\boldsymbol{j}}=\bar{\boldsymbol{I}}}^{\bar{\boldsymbol{N}}} \boldsymbol{\varphi}_{\bar{\boldsymbol{j}}}(\bar{\boldsymbol{x}}) \boldsymbol{C}\left(\bar{\boldsymbol{x}}_{\bar{\boldsymbol{j}}}\right) \tag{19}$$

where

$\boldsymbol{\varphi}_{\bar{\boldsymbol{j}}}$ = characteristic function of the hyper-volume $\boldsymbol{\Omega}_{\bar{\boldsymbol{j}}}$ surrounding the node $\bar{\boldsymbol{x}}_{\bar{\boldsymbol{j}}}$:

$$\boldsymbol{\varphi}_{\bar{\boldsymbol{j}}}(\bar{\boldsymbol{x}}) = \begin{cases} \boldsymbol{1}, \boldsymbol{if}\ \bar{\boldsymbol{x}} \in \boldsymbol{\Omega}_{\bar{\boldsymbol{j}}} \\ \boldsymbol{0}, \boldsymbol{if}\ \bar{\boldsymbol{x}} \notin \boldsymbol{\Omega}_{\bar{\boldsymbol{j}}} \end{cases} \tag{20}$$

where

$$\boldsymbol{\Omega}_{\bar{\boldsymbol{j}}} = \prod_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{D}} \left[\boldsymbol{x}_{\boldsymbol{j}_i} - \frac{\boldsymbol{h_i}}{\boldsymbol{2}}, \boldsymbol{x}_{\boldsymbol{j}_i} + \frac{\boldsymbol{h_i}}{\boldsymbol{2}} \right] \tag{21}$$

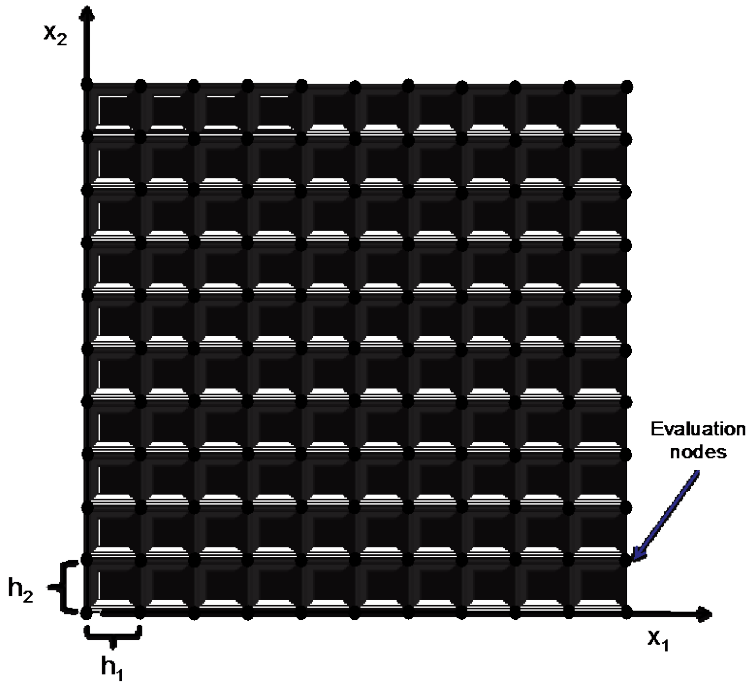

Figure 12. Discretization grid.

The probability of the uncertain parameters is expressed as:

$$\boldsymbol{pdf}_{\bar{X}}(\bar{x}) = \sum_{\bar{j}=\bar{1}}^{\bar{N}} \boldsymbol{\varphi}_{\bar{j}}(\bar{x}) \boldsymbol{pdf}_{\bar{X}}(\bar{x}_{\bar{j}}) \tag{22}$$

Following the approach briefly explained in Subsection 4.1, the probability of the event (e.g., failure) could be expressed as:

$$\boldsymbol{P_F} = \left(\prod_{i=1}^{D} \boldsymbol{h_i} \right) \left(\sum_{\bar{j}=\bar{1}}^{\bar{N}} \boldsymbol{pdf}_{\bar{X}}(\bar{x}_{\bar{j}}) \boldsymbol{C}(\bar{x}_{\bar{j}}) \right) \tag{23}$$

Under certain assumptions [18], the concept of active hyper-volume $\boldsymbol{V_A}$ as the region of the input space identified by the support of the uncertain parameters' probability density functions $\boldsymbol{pdf}_{\bar{X}}(\bar{x})$ could be introduced; *Equation (23)* is re-casted, using a Taylor expansion, as follows:

$$\boldsymbol{P_F} = \int_{V} \boldsymbol{C}(\bar{x}) \boldsymbol{pdf}_{\bar{X}}(\bar{x}) d\bar{x} = \int_{V_A} \boldsymbol{C}(\bar{x}) \left[\sum_{\bar{j}=\bar{1}}^{\bar{N}} \boldsymbol{\varphi}_{\bar{j}}(\bar{x}) \left(\boldsymbol{pdf}_{\bar{X}}(\bar{x}_{\bar{j}}) + \sum_{i=1}^{D} \frac{\partial \boldsymbol{pdf}_{\bar{X}}}{\partial \boldsymbol{x_i}} \Big|_{\bar{x}_{\bar{j}}} (\boldsymbol{x_i} - \boldsymbol{x}_{j_i}) \right) \right] d\bar{x} \tag{24}$$

And, considering the evaluation grid as:

$$\boldsymbol{P_F} = \sum_{\substack{\bar{j}=\bar{1} \\ \bar{x}_{\bar{j}} \in \boldsymbol{V_A}}}^{\bar{N}} \int_{\bar{x}_{\bar{j}}-\bar{h}/2}^{\bar{x}_{\bar{j}}-\bar{h}/2} \boldsymbol{C}(\bar{x}) \left(\boldsymbol{pdf}_{\bar{X}}(\bar{x}_{\bar{j}}) + \sum_{i=1}^{D} \frac{\partial \boldsymbol{pdf}_{\bar{X}}}{\partial \boldsymbol{x_i}} \Big|_{\bar{x}_{\bar{j}}} (\boldsymbol{x_i} - \boldsymbol{x}_{j_i}) \right) d\bar{x} \tag{25}$$

At this point, it is possible to label, in the active hyper-volume, the subdomain identified by the nodes where the "goal" function $\boldsymbol{C}(\bar{x})$ changes its value (the frontier nodes between the region where $\boldsymbol{C}(\bar{x}) = \boldsymbol{1}$ and $\boldsymbol{C}(\bar{x}) = \boldsymbol{0}$) $\boldsymbol{V_A} \cap \boldsymbol{V}_{\boldsymbol{\partial V_F}}$.

Consequentially, it is possible to identify the subdomains in which the "goal" function $\boldsymbol{C}(\bar{x})$ is equal to 0 ($\boldsymbol{V_A} \cap \boldsymbol{V}_{\boldsymbol{C}(\bar{x})=\boldsymbol{0}} \notin \boldsymbol{V_A} \cap \boldsymbol{V}_{\boldsymbol{\partial V_F}}$) and 1 ($\boldsymbol{V_A} \cap \boldsymbol{V}_{\boldsymbol{C}(\bar{x})=\boldsymbol{1}} \notin \boldsymbol{V_A} \cap \boldsymbol{V}_{\boldsymbol{\partial V_F}}$):

$$\sum_{\substack{\bar{j}=\bar{1} \\ \bar{x}_{\bar{j}} \in \boldsymbol{V_A} \cap \boldsymbol{V}_{\boldsymbol{C}(\bar{x})=\boldsymbol{0}}}}^{\bar{N}} \int_{\bar{x}_{\bar{j}}-\bar{h}/2}^{\bar{x}_{\bar{j}}-\bar{h}/2} \boldsymbol{C}(\bar{x}) \left(\boldsymbol{pdf}_{\bar{X}}(\bar{x}_{\bar{j}}) + \sum_{i=1}^{D} \frac{\partial \boldsymbol{pdf}_{\bar{X}}}{\partial \boldsymbol{x_i}} \Big|_{\bar{x}_{\bar{j}}} (\boldsymbol{x_i} - \boldsymbol{x}_{j_i}) \right) d\bar{x} = \boldsymbol{0} \tag{26}$$

$$\begin{aligned} &\sum_{\substack{\bar{j}=\bar{1} \\ \bar{x}_{\bar{j}} \in \boldsymbol{V_A} \cap \boldsymbol{V}_{\boldsymbol{C}(\bar{x})=\boldsymbol{1}}}}^{\bar{N}} \int_{\bar{x}_{\bar{j}}-\bar{h}/2}^{\bar{x}_{\bar{j}}-\bar{h}/2} \boldsymbol{C}(\bar{x}) \left(\boldsymbol{pdf}_{\bar{X}}(\bar{x}_{\bar{j}}) + \sum_{i=1}^{D} \frac{\partial \boldsymbol{pdf}_{\bar{X}}}{\partial \boldsymbol{x_i}} \Big|_{\bar{x}_{\bar{j}}} (\boldsymbol{x_i} - \boldsymbol{x}_{j_i}) \right) d\bar{x} \\ &= \sum_{\substack{\bar{j}=\bar{1} \\ \bar{x}_{\bar{j}} \in \boldsymbol{V_A} \cap \boldsymbol{V}_{\boldsymbol{C}(\bar{x})=\boldsymbol{1}}}}^{\bar{N}} \int_{\bar{x}_{\bar{j}}-\bar{h}/2}^{\bar{x}_{\bar{j}}-\bar{h}/2} \left(\boldsymbol{pdf}_{\bar{X}}(\bar{x}_{\bar{j}}) + \sum_{i=1}^{D} \frac{\partial \boldsymbol{pdf}_{\bar{X}}}{\partial \boldsymbol{x_i}} \Big|_{\bar{x}_{\bar{j}}} (\boldsymbol{x_i} - \boldsymbol{x}_{j_i}) \right) d\bar{x} \end{aligned} \tag{27}$$

Equation (25) is now expressed as:

$$\boldsymbol{P_F} = \sum_{\substack{\bar{j}=\bar{1} \\ \bar{x}_{\bar{j}} \in \boldsymbol{V_A} \cap \boldsymbol{V_{C(\bar{x})=1}}}}^{\bar{N}} \left(\prod_{i=1}^{D} \boldsymbol{h_i} \right) \boldsymbol{pdf}_{\bar{X}}(\bar{x}_{\bar{j}}) + \boldsymbol{O}(\boldsymbol{h}^{N+1}) + \sum_{\substack{\bar{j}=\bar{1} \\ \bar{x}_{\bar{j}} \in \boldsymbol{V_A} \cap \boldsymbol{V_{\partial V_F}}}}^{\bar{N}} \int_{\bar{x}_{\bar{j}}-\bar{h}/2}^{\bar{x}_{\bar{j}}-\bar{h}/2} \boldsymbol{C}(\bar{x}) \left(\boldsymbol{pdf}_{\bar{X}}(\bar{x}_{\bar{j}}) + \sum_{i=1}^{D} \frac{\partial \boldsymbol{pdf}_{\bar{X}}}{\partial \boldsymbol{x_i}} \Big|_{\bar{x}_{\bar{j}}} (\boldsymbol{x_i} - \boldsymbol{x_{j_i}}) \right) \boldsymbol{d\bar{x}} \tag{28}$$

As inferred from *Equation (28)*, the process is bounded if the surface-area-to-volume ratio (amount of surface area per unit volume) is in favor of the volume:

$$\sum_{\substack{\bar{j}=\bar{1} \\ \bar{x}_{\bar{j}} \in \boldsymbol{V_A} \cap \boldsymbol{V_{C(\bar{x})=1}}}}^{\bar{N}} \left(\prod_{i=1}^{D} \boldsymbol{h_i} \right) \boldsymbol{pdf}_{\bar{X}}(\bar{x}_{\bar{j}}) \gg \sum_{\substack{\bar{j}=\bar{1} \\ \bar{x}_{\bar{j}} \in \boldsymbol{V_A} \cap \boldsymbol{V_{\partial V_F}}}}^{\bar{N}} \left| \int_{\bar{x}_{\bar{j}}-\frac{\bar{h}}{2}}^{\bar{x}_{\bar{j}}-\frac{\bar{h}}{2}} \boldsymbol{pdf}_{\bar{X}}(\bar{x}_{\bar{j}}) \boldsymbol{d\bar{x}} \right| \tag{29}$$

If the grid is built in the transformed space of probability (i.e., replacing the measure $\boldsymbol{d\bar{x}}$ with $\boldsymbol{d\bar{\mu}} = \boldsymbol{pdf}_{\bar{X}} \boldsymbol{d\bar{x}}$), the condition expressed in *Equation (29)* is reduced:

$$\#\,\boldsymbol{nodes} \in \boldsymbol{V_A} \cap \boldsymbol{V_{C(\bar{x})=1}} \gg \#\,\boldsymbol{nodes} \in \boldsymbol{V_A} \cap \boldsymbol{V_{\partial V_F}} \tag{30}$$

This means that error is bounded by the total probability contained in the cells on the frontier of the LS.

Based on this derivation, it is clear how important it is to keep the content of the total probability on the frontier of the LS as low as possible, and simultaneously, increase the importance of the volume of the failure/event region as much as possible (to improve the surface-area-to-volume ratio).

In order to do that, the step size in probability should be significantly reduced ($\boldsymbol{h_i^p} \to \boldsymbol{0^+}$). Even if this is theoretically feasible, it is computational inapplicable. To approach a similar result, it is possible to learn from other numerical methods that use the technique of adaptive meshing for the resolution of the partial differential equation system (e.g., finite element methods).

For this reason, an acceleration scheme was designed and developed employing a multi-grid approach. The main idea, it is to re-cast the iterative process in two different subsequential steps. Firstly, performing the LS search on a coarse evaluation grid, and once converged, adaptively refining the cells that lie on the frontier of the LS ($\boldsymbol{V_A} \cap \boldsymbol{V_{\partial V_F}}$) and, consequentially, converging on the new refined grid.

The iteration scheme is graphically shown in Figure 13.

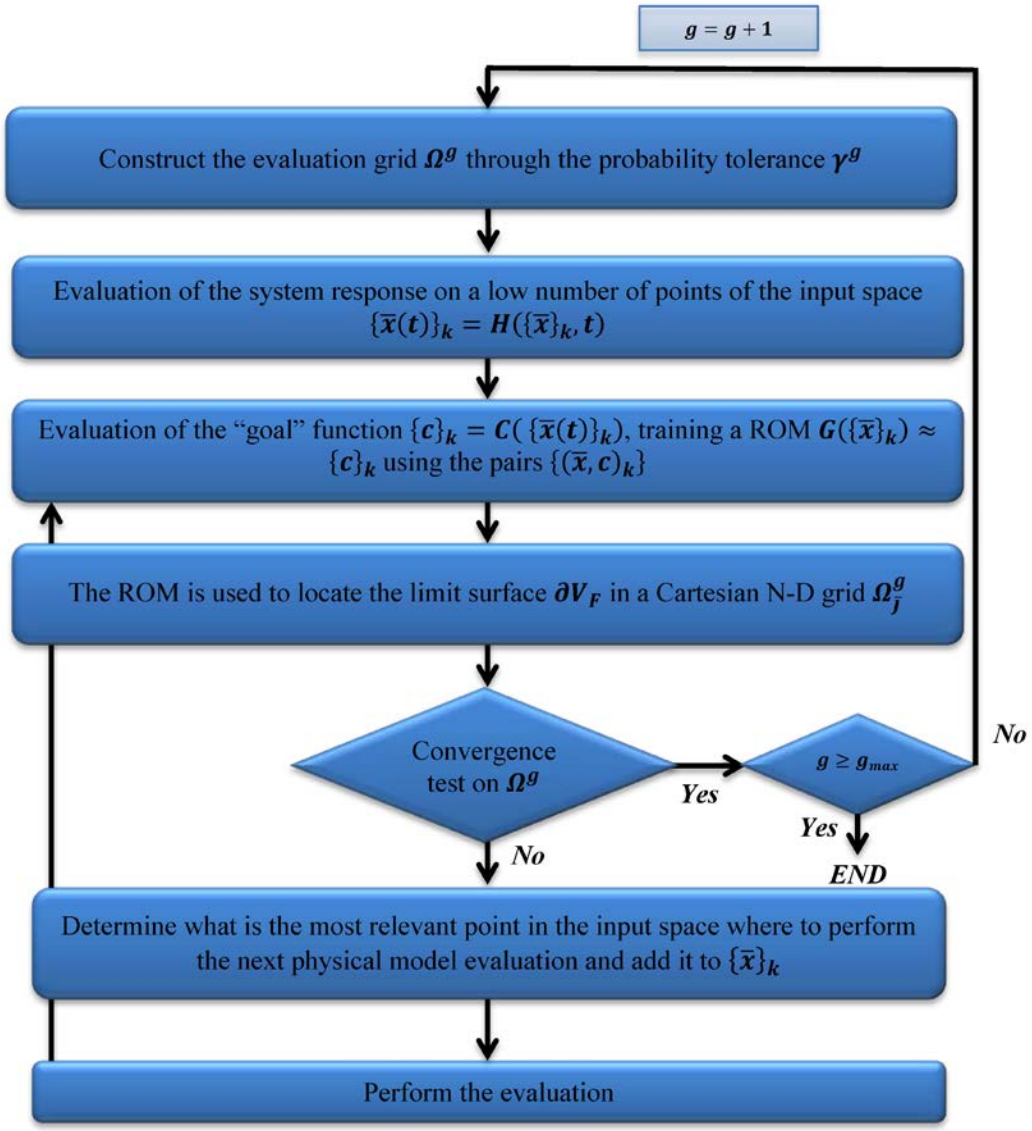

Figure 13. Multi-grid limit surface search scheme.

In more detail, the iterative scheme could be summarized through the following steps:

1. The user specifies two tolerances in probability (***CDF***): $\gamma^{g=1}$ for the initial coarse grid and $\gamma^{g=2}$ for the refined grid, where $\gamma^{g=1} > \gamma^{g=2}$.
2. Following *Equation (21)*, the initial coarse evaluation grid Ω^1 is constructed ($N^{g=1}$ total nodes). The discretization of this grid is done to have cells with a content of probability equal to $\gamma^{g=1}$.
3. A limited number of points in the uncertain space $\{\bar{x}_k\}$ are selected via one of the forward sampling strategies (e.g., stratified or MC).
4. The high-fidelity code is used to compute the status of the system for the set of points in the input set: $\{\bar{x}(t)\}_k = H(\{\bar{x}\}_k, t)$.
5. The "goal" function is evaluated at the phase space coordinate of the system: $\{c\}_k = C(\{\bar{x}(t)\}_k)$.
6. The set of pairs $\{(\bar{x}, c)_k\}$ are used to train a ROM of type classifier, $G(\{\bar{x}\}_k)$.

7. The ROM classifier is used to predict the values of the "goal" function for all the $\boldsymbol{N}$ nodes of the N-D grid in the domain space: $\boldsymbol{G}\left(\{\bar{\boldsymbol{x}}\}_{\boldsymbol{j}}\right) \approx \{\boldsymbol{c}\}_{\boldsymbol{j}}, \boldsymbol{j} = \boldsymbol{1}, \dots, \boldsymbol{N^{g=1}}$.
8. The values of the "goal" function are used to determine the LS location based on the change of values of $\{\boldsymbol{c}\}_{\boldsymbol{j}} \rightarrow \boldsymbol{\partial V_F}$.
9. A new point is chosen to increase the training set and a new pair is generated.
10. The procedure is repeated starting from Step 5 until convergence is achieved on grid $\boldsymbol{\Omega^g}$. The convergence is reached when there are no changes in the location of the LS after a certain number of consecutive iterations (user-defined).
11. When the convergence is achieved on the coarse grid $\boldsymbol{\Omega^{g=1}}$, all the cells that lie on the frontier of the LS ($\boldsymbol{V_A} \cap \boldsymbol{V_{\partial V_F}}$) are refined to contain an amount of probability equal to $\boldsymbol{\gamma^{g=2}}$.
12. Steps 7 through 9 are performed based on the new refined grid. Finally, the process starts again by performing Steps 5 through 10, until the convergence is achieved in the refined grid.

As shown in Figure 13, the algorithm consists in searching the location of the LS proceeding with subsequential refinement of the subdomain, in the active space, that contains the LS. In this way, the computational burden is kept as low as possible. In addition, another advantage of this approach is that, since the refinement grid represents a constrained domain, the subsequential ROM training process can be regularized, since the LS between an iteration and the other can move, at maximum, within the refinement domain.

5. TEST CASES

To test the validity of the acceleration schemes, some test cases were designed and employed. For all test cases, the LS search is performed employing the algorithm that was presented in the June 2014 milestone, the new acceleration schemes and a brute force approach where an MC sampling is used to generate a set of data that are used for a posteriori construction of the LS.

It is worth mentioning that, even if in all the test cases presented hereafter, the independent variables defining the input space are associated with PDFs this is not required for application of the LS concept and the search methodologies presented in this report.

When a probability is not associated with the input space, the LS search is simply a parametric search of the input space where the system satisfies certain constrains (e.g., does not fail under the accident scenario considered).

The reason why all tests presented in this report possess a stochastic characterization of the input space is because one of the comparison metrics, between the different search methodologies, taken in account is the probability of the binary event represented by the "goal" function (e.g., the probability of failure).

In the following subsections, three different examples are presented; the first two are based on analytical models using the RAVEN "external model" API, presented in Subsection 3.2.2.3, and the third is based on a pressurized water reactor (PWR) SBO employed with RELAP-7.

5.1 Simple Three-Dimensional Analytical Test

As previously mentioned, the first test was performed using the RAVEN "external model." The response of the system is modeled with the following simple equation:

$$y = {x_1}^2 + x_1 * x_2 * x_3 \tag{31}$$

where

y = outcome of the system

$x_1, x_2,$ and x_3 = independent variables.

It is assumed that the independent variables are affected by uncertainties that are stochastically modeled as follows:

$$X_1 \sim N(\mu_1, \sigma_1);\ pdf(X_1 = x_1) = \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{(x_1-\mu_1)^2}{2\sigma_1^2}} \quad 0.0 \le x_1 \le 1.0 \quad \mu_1 = 0.5 \quad \sigma_1 = 0.1 \tag{32}$$

$$X_2 \sim N(\mu_2, \sigma_2);\ pdf(X_2 = x_2) = \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{(x_2-\mu_2)^2}{2\sigma_2^2}} \quad 0.0 \le x_2 \le 4.0 \quad \mu_2 = 2.0 \quad \sigma_2 = 0.2 \tag{33}$$

$$X_3 \sim U(a, b);\ pdf(X_3) = \begin{cases} \frac{1}{b-a} & a \le x_3 \le b \\ 0 & x_3 < a,\ x_3 > b \end{cases} \quad a = 1.0 \quad b = 4.0 \tag{34}$$

For this particular case, the "goal" function represents a threshold phenomenon:

$$C(y) = \begin{cases} +1 \ if\ y > 5.0 \\ -1 \ if\ y \le 5.0 \end{cases} \tag{35}$$

To compare the new multi-grid acceleration scheme and previous approach, the convergence criteria in Table 1 are used.

Table 1. Simple three-dimensional analytical test: Limit surface search convergence criteria.

Parameter	Multi-grid LS Search	Fixed-grid LS Search
Persistency	25	25
Tolerance (CDF)	2.7E−5	1.E−6*
Multi-grid tolerance (CDF)	1.E−6*	—
*Final probability convergence tolerance.		

The ROM used for accelerating the convergence is an SVM classifier with a kerned based on radial basis function.

In addition, a 100.000 samples MC is run; subsequentially, the a-posteriori LS is computed.

For all three employed sampling strategies, the final probability is computed using the LS integral post-processor explained in Subsection 4.4.1.

Figure 14 and Figure 15 show the location of the samples and final LS for the multi-grid approach. There are no figures shown for the LS search method as they are similar to figures shown for the previous LS search method.

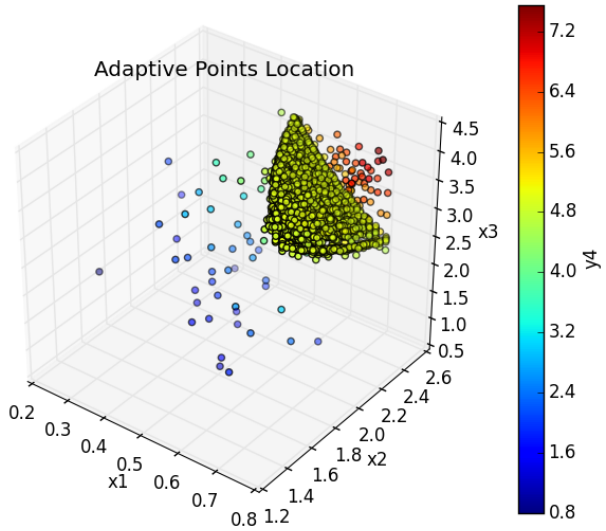

Figure 14. Samples' location in multi-grid limit surface approach (three-dimensional analytical test).

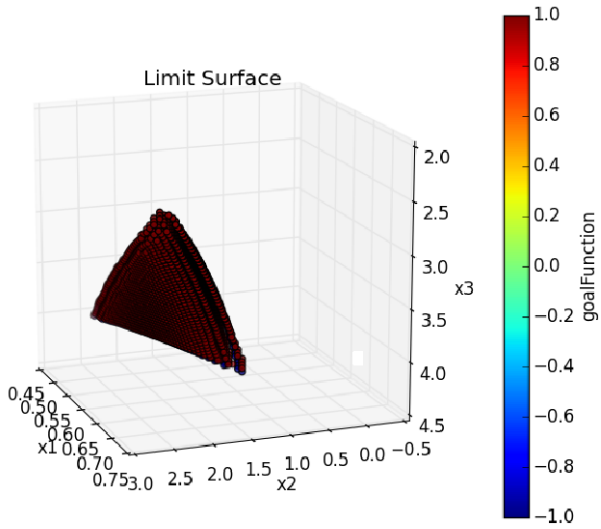

Figure 15. Limit surface in multi-grid limit surface approach (three-dimensional analytical test).

Table 2 compares the multi-grid and fixed-grid approaches. It is noticeable that the new acceleration scheme is brought to a speed-up factor (in terms of number of iterations and, consequentially, of runs) of approximately 2.6 (i.e., the multi-grid approach converged 2.6 times faster than the fixed-grid one).

Table 2. Multi-grid and fixed-grid comparison (three-dimensional analytical test).

Parameter	Multi-grid LS Search	Fixed-grid LS Search	Speed-up Factor
Number of iterations	1,951	5,085	Approximately 2.6
Probability	2.8894E−02	2.8894E−02	—
Probability MC	3.2750E−02	—	—

The probability computed with the LS obtained for both approaches is the same; this is another indication that both methodologies converge on the same solution (the converged LS location is exactly the same).

5.2 Two-Dimensional Test Case in Presence of Failure Islands

In determining failure boundaries, the most challenging problem is identifying failure regions that are represented by a convex set of points and, thus, that completely isolate a portion of the uncertain domain. In safety analysis and PRA analysis, these types of boundaries are named failure "islands." To test the new multi-grid algorithm with such a challenging problem, another analytical test was designed, using, again, the RAVEN "external model" API. The response of the system is modeled with the following simple equation:

$$\boldsymbol{y = x_1^2 + x_2^2} \tag{36}$$

where

$\boldsymbol{y}$ = outcome of the system

$\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ = independent variables.

It is assumed that the independent variables are affected by uncertainties that are stochastically modeled as follows:

$$\boldsymbol{X_1 \sim N(\mu_1, \sigma_1);\ pdf(X_1 = x_1) = \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{(x_1-\mu_1)^2}{2\sigma_1^2}} \quad -5.0 \le x_1 \le 5.0 \quad \mu_1 = 0.0 \quad \sigma_1 = 1.5} \tag{37}$$

$$\boldsymbol{X_2 \sim N(\mu_2, \sigma_2);\ pdf(X_2 = x_2) = \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{(x_2-\mu_2)^2}{2\sigma_2^2}} \quad -6.0 \le x_2 \le 6.0 \quad \mu_2 = 0.0 \quad \sigma_2 = 1.2} \tag{38}$$

For this particular case, the "goal" function represents a two-threshold phenomenon:

$$\boldsymbol{C(y) = \begin{cases} +1 & if\ 0.5 < y \le 1.0 \\ -1 & if\ y \le 5.0 \end{cases}} \tag{39}$$

To compare the new multi-grid acceleration scheme and previous approach, the convergence criteria in Table 3 are used.

Table 3. Two-dimensional analytical test in presence of failure islands: Limit surface search convergence criteria.

Parameter	Multi-grid LS Search	Fixed-grid LS Search
Persistency	20	20
Tolerance (CDF)	5.625E−6	6.25E−7*
Multi-grid tolerance (CDF)	6.25E−7*	—
*Final probability convergence tolerance.		

The ROM used for accelerating the convergence is an SVM classifier with a kerned based on radial basis function.

In addition, a 100.000 samples MC is run; subsequentially, the a-posteriori LS is computed.

For all three employed sampling strategies, the final probability is computed using the LS integral post-processor mentioned in Subsection 4.4.1.

Figure 16 and Figure 17 show the location of the samples and final LS for the multi-grid approach. There are no figures shown for the LS search method as they are similar to figures shown for the previous LS search method.

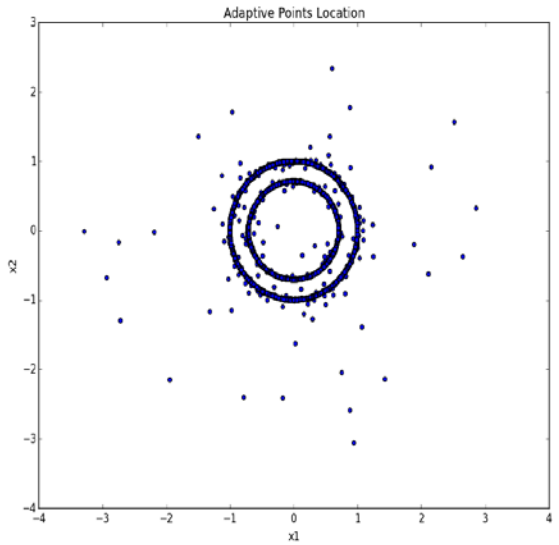

Figure 16. Samples' location in multi-grid limit surface approach (two-dimensional analytical test in presence of failure islands).

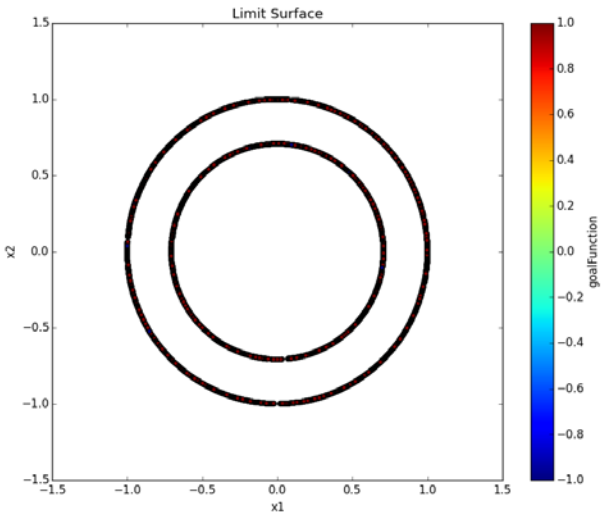

Figure 17. Limit surface in multi-grid limit surface approach (two-dimensional analytical test in presence of failure islands).

Table 4 compares the multi-grid and fixed-grid approaches. The number of iterations for both methodologies testifies how challenging the identification of failure "islands" is. Anyhow, it is noticeable that the new acceleration scheme is brought to a speed-up factor of approximately2.4 (i.e., the multi-grid approach converged 2.4 times faster than the fixed-grid one).

Table 4. Multi-grid and fixed-grid comparison (two-dimensional analytical test in presence of failure islands).

Parameter	Multi-grid LS Search	Fixed-grid LS Search	Speed-up Factor
Number of iterations	5,300	12,354	Approximately 2.4
Probability	1.1241E−01	1.1241E−01	—
Probability MC	1.1253E−01	—	—

The probability computed with the LS obtained for both approaches is the same; this is another indication that both methodologies converge on the same solution (the converged LS location is exactly the same). In addition, the difference between the probabilities computed with the MC approach and LS search approach is within the 0.1%.

5.3 Pressurized Water Reactor Station Black-out Demo Using RELAP-7 as System Code

The previous examples demonstrated the functionality of the multi-grid LS search on analytical tests. It is important to assess how the new algorithm reacts to a real safety case. For this purpose, a PWR model was built, at INL, in the thermal-hydraulic code RELAP-7. The model is set up based on the parameters specified in the Organization for Economic Cooperation and Development main steam line break benchmark problem. [19] The reference design for the Organization for Economic Cooperation and Development main steam line break benchmark problem is derived from the reactor geometry and operational data of the Three Mile Island-1 NPP, which is a 2,772-MW, two-loop PWR (see the system scheme shown in Figure 18).

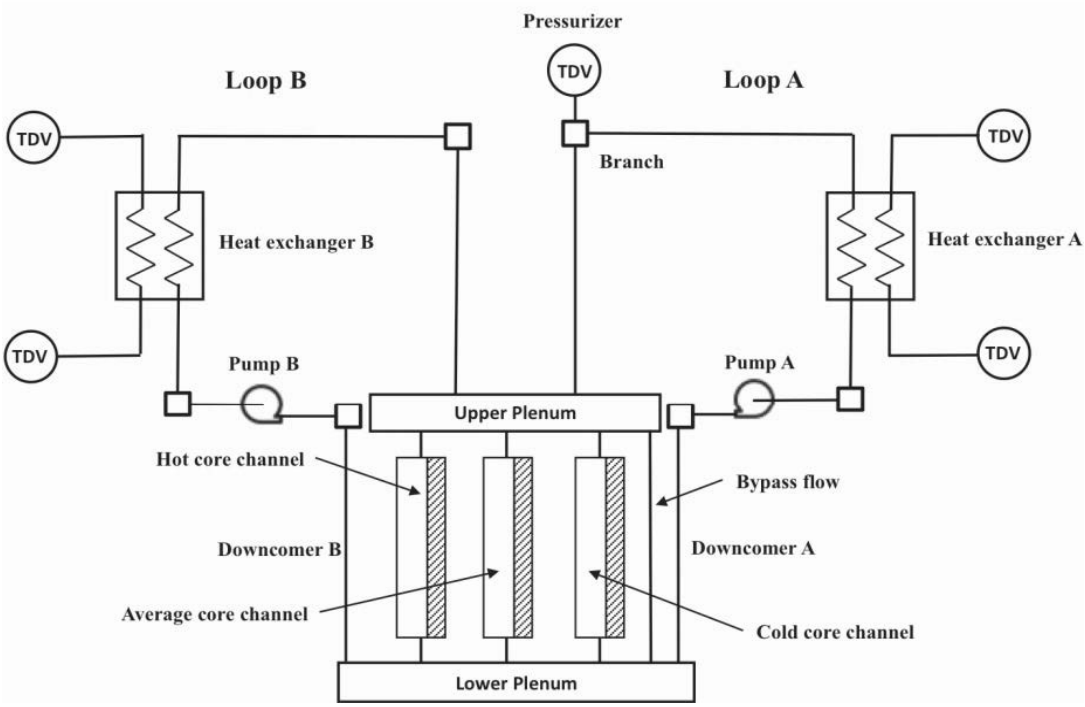

Figure 18. Scheme of Three Mile Island pressurized water reactor benchmark.

The simulated scenario is a simplified SBO accident.

To reach a steady-state condition, the simulation is run for 500 seconds without any change in its internal parameters. The reference scenario is summarized as follows (see Figure 19):

- At $t = 500\ s$, the external initiating event (e.g., earthquake) causes a loss of outside power event. The reactor successfully scrams, AC power is provided by the DGs and the ECCS keeps the reactor core cool.
- At $t_1 = 2000\ s$, the DGs, which were providing emergency AC power, become unavailable. Without AC power, the ECCS is disabled as well and the core temperature increases. When AC power is recovered (t_2), through either DGs or primary grid recovery, ECCS capabilities are restored and core temperature starts to decrease.

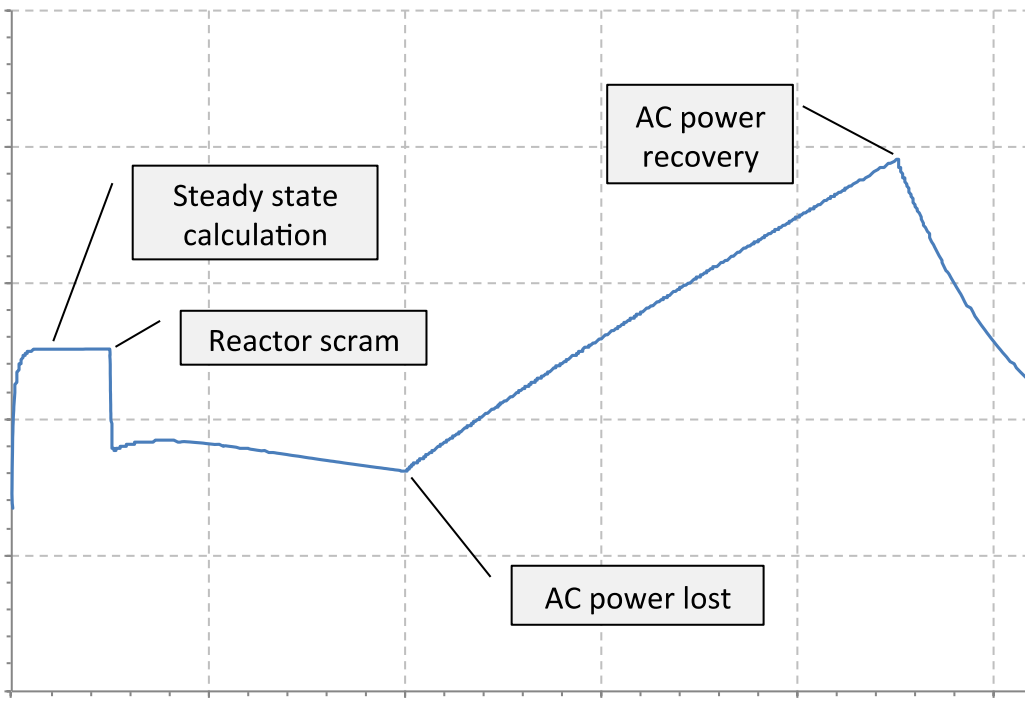

Figure 19. Example of loss of outside power scenario followed by diesel generators' failure using RELAP-7 code.

It is assumed that the DGs' failure and ECCS recovery times are affected by uncertainties that are stochastically modeled as follows:

$$\boldsymbol{X_1 \sim N(\mu_1, \sigma_1);\ pdf(X_1 = x_1) = \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{(x_1-\mu_1)^2}{2\sigma_1^2}} \quad 0.0 \le x_1 \le 8{,}000.0 \quad \mu_1 = 4{,}000.0 \quad \sigma_1 = 1{,}000.0} \tag{40}$$

$$\boldsymbol{X_2 \sim N(\mu_2, \sigma_2);\ pdf(X_2 = x_2) = \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{(x_2-\mu_2)^2}{2\sigma_2^2}} \quad 0.0 \le x_2 \le 40{,}000.0 \quad \mu_2 = 1800.0 \quad \sigma_2 = 4{,}000.0} \tag{41}$$

For this particular case, the "goal" function models a threshold on the peak-clad temperature (the maximum temperature reached by the cladding). For this analysis, a threshold of 1477.6 K (2200°F) is used:

$$\boldsymbol{C(y) = \begin{cases} +1\ (failure)\ \ if\ T_{clad} \ge 1477.6\ K \\ -1\ (success)\ \ if\ T_{clad} < 1477.6\ K \end{cases}} \tag{42}$$

To compare the new multi-grid acceleration scheme and previous approach, the convergence criteria in Table 5 are used.

Table 5. RELAP-7 station black-out analysis: Limit surface search convergence criteria.

Parameter	Multi-grid LS Search	Fixed-grid LS Search
Persistency	20	20
Tolerance (CDF)	5.625E−6	6.25E−7*
Multi-grid tolerance (CDF)	6.25E−7*	—
*Final probability convergence tolerance.		

Also for this case, an SVM classifier, with a radial basis function kernel, is used in the LS search process.

In addition, a 100.000 samples MC is run; subsequentially, the a-posteriori LS is computed.

As for the other cases, for all three employed sampling strategies, the final probability is computed using the LS integral post-processor explained in Subsection 4.4.1.

Figure 20 and Figure 21 show the location of the samples and final LS for the multi-grid approach. There are no figures shown for the LS search method as they are similar to figures shown for the previous LS search method.

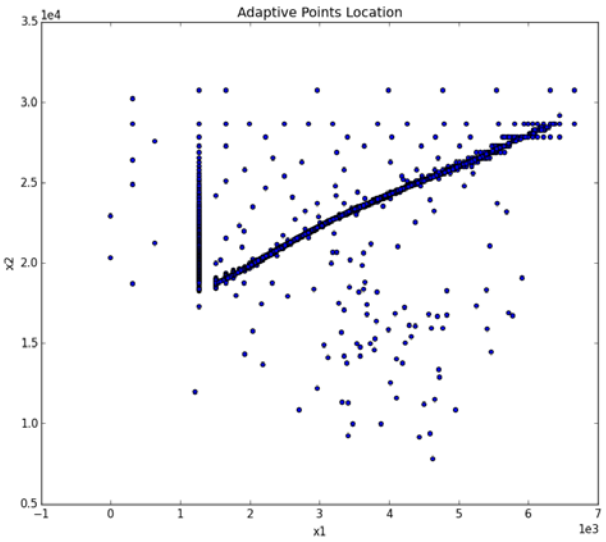

Figure 20. Samples' location in multi-grid limit surface approach (two-dimensional pressurized water reactor station black-out scenario).

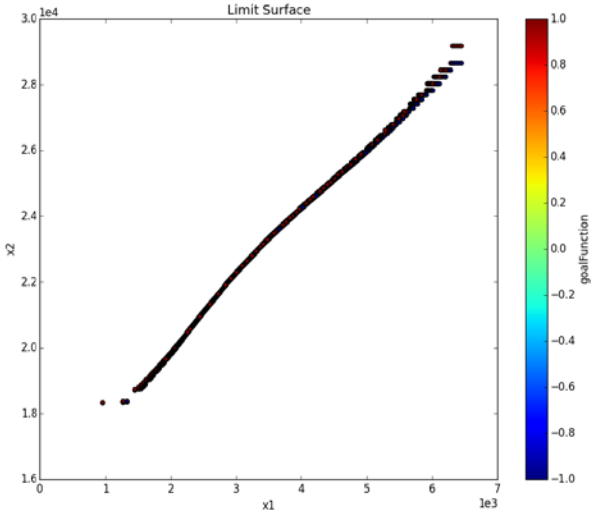

Figure 21. Limit surface in multi-grid limit surface approach (loss of outside power scenario).

Table 6 compares the multi-grid and fixed-grid approaches.

Table 6. Multi-grid and fixed-grid comparison: RELAP-7 station black-out analysis.

Parameter	Multi-grid LS Search	Fixed-grid LS Search	Speed-up Factor
Number of iterations	1701	2902	Approximately 1.7
Probability	8.4170E−02	8.4170E−02	—
Probability MC	8.4254E−02	—	—

It is noticeable that determination of the LS location, in this case, is quite trivial. In such case, the gain in the convergence is not as good as the previous tests. Anyhow, a gain factor of approximately 1.7, for high-demanding physical models (e.g., system codes), can consistently decrease the number of CPU hours needed for a full PRA analysis.

The probability computed with the LS obtained for both approaches is the same; this is another indication that both methodologies converge on the same solution (the converged LS location is exactly the same). In addition, the difference between the probabilities computed with the MC approach and LS search approach is within the 0.1%.

6. FUTURE DEVELOPMENT

In this report, a multi-grid acceleration scheme was explained. During accomplishment of this work, other possible methodologies were identified for the improvement of the convergence of the LS search. As reported in Subsection 4.4.2, the multi-grid approach consists in a two-step iterative process, firstly converging on a coarse grid and, then, on a refined one. This approach lets the user choose the refinement strategy, that most of the times, is quite complicated to define, overall for the stability of the ROM. To overcome and simplify the use of this new algorithm, in the near future, the method will be improved by adding an automated refinement scheme to focalize in the stabilization of the synthetic operator represented by the ROM.

Another path of research and development is about the wrapping of the ROMs in an “outer” iterative process, based on the optimization of their internal parameters (e.g., penalty error factors and smoothing parameters). This converged ROM will then be used as an acceleration algorithm for the LS search method.

In addition, development will focus on identifying and employing better algorithms for efficiently choosing the points in the phase space that need explored. The authors would like to explore a methodology that is based on using the directional derivatives directly computed, during the iterative process, on the LS.

7. CONCLUSIONS

This report focused on new development of the acceleration of the LS search from a mathematical and numerical point of view. The new acceleration algorithm based on the multi-grid approach looks extremely promising. It significantly improved the convergence of the LS search process, simultaneously reducing the CPU time to reach a stable solution.

To design and develop the method, several developments were necessary; these developments, only briefly addressed in this report, are crucial for further research and development activities around the LS search process and will significantly facilitate the future implementations, allowing the RAVEN team to faster deploy future new acceleration schemes.

Overall, RAVEN is proposed as a valid tool for a more comprehensive and computational efficient PRA analysis.

8. REFERENCES

1. Rabiti, C., A. Alfonsi, D. Mandelli, J. Cogliati, and R. Kinoshita, "RAVEN, a New Software for Dynamic Risk Analysis," PSAM 12 Probabilistic Safety Assessment and Management, Honolulu, Hawaii, June 2014.
2. Rabiti, C., A. Alfonsi, D. Mandelli, J. Cogliati, R. Martinueau, and C. Smith, *Deployment and Overview of RAVEN Capabilities for a Probabilistic Risk Assessment Demo for a PWR Station Blackout*, INL/EXT-13-29510, Idaho National Laboratory, 2013.
3. Rabiti, C., A. Alfonsi, D. Mandelli, J. Cogliati, R. Kinoshita, and S. Sen, *RAVEN User Manual*, INL/EXT-15-34123, Idaho National Laboratory, 2015.
4. Alfonsi, A., C. Rabiti, D. Mandelli, J. Cogliati, R. Kinoshita, and A. Naviglio, "RAVEN and dynamic probabilistic risk assessment: Software overview," in Proceedings of ESREL European Safety and Reliability Conference, 2014.
5. Alfonsi, A., C. Rabiti, D. Mandelli, J. Cogliati, and R. Kinoshita, "Raven as a tool for dynamic probabilistic risk assessment: Software overview," in Proceedings of M&C2013 International Topical Meeting on Mathematics and Computation, CD-ROM, American Nuclear Society, LaGrange Park, IL, 2013.
6. Rabiti, C., 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, ID, pp. 320–332, 2013.
7. Smith, C., C. Rabiti, and R. Martineau, *Risk Informed Safety Margins Characterization (RISMC) Pathway Technical Program Plan*, INL/EXT-11-22977, Idaho National Laboratory, 2011.
8. *Light Water Reactor Sustainability Program Integrated Program Plan*, INL-EXT-11-23452, Idaho National Laboratory, 2011.
9. Rabiti, C., A. Alfonsi, D. Mandelli, J. Cogliati, and B. Kinoshita, *Advanced Probabilistic Risk Analysis Using RAVEN and RELAP-7*, INL/EXT-14-32491, Idaho National Laboratory, 2014.
10. David, A., R. Berry, D. Gaston, R. Martineau, J. Peterson, H. Zhang, H. Zhao, and L. Zou, *RELAP-7 Level 2 Milestone Report: Demonstration of a Steady State Single Phase PWR Simulation with RELAP-7*, INL/EXT-12-25924, Idaho National Laboratory, 2012.
11. Habermann, C., and F. Kindermann, "Multidimensional Spline Interpolation: Theory and Applications," *Computational Economics*, Volume 30, Issue 2, pp. 153-169.
12. Gordon, W.J., and J.A.Wixom, "Shepard's Method of Metric Interpolation to Bivariate and Multivariate Interpolation," *In Mathematics and Computation*, Volume 32, Issue 141, pp. 253-264, 1978.
13. Rabiti, C., P. Talbot, A. Alfonsi, D. Mandelli, and J. Cogliati, *Implementation of Stochastic Polynomials Approach in the RAVEN Code*, INL/EXT-13-30611, Idaho National Laboratory, 2013.
14. Alfonsi, A., C. Rabiti, D. Mandelli, J. Cogliati, R. Kinoshita, and A. Naviglio, "Dynamic event tree analysis through Raven," in Proceedings of ANS PSA 2013 International Topical Meeting on Probabilistic Safety Assessment and Analysis, 2013.
15. RELAP5 Code Development Team, RELAP5-3D Code Manual, Idaho National Laboratory, 2012.
16. Gaston, D., C. Newman, G. Hansen, and D. Lebrun-Grandi, "MOOSE: A parallel computational framework for coupled systems of nonlinear equations," *Nuclear Engineering Design*, 239, pp. 1768-1778, 2009.

17. Alfonsi, A., C. Rabiti, A. S. Epiney, Y. Wang, and J. Cogliati, “PHISICS Toolkit: Multi-Reactor Transmutation Analysis Utility–MRTAU,” in Proceedings of PHYSOR 2012 “Advances in Reactor Physics Linking Research, Industry, and Education,” Knoxville, TN, April 15-20, 2012.

18. Rabiti, C., J. Cogliati, G Pastore, R. J Gardner, and A. Alfonsi, “Fuel reliability analysis using Bison and Raven,” in Proceedings of ANS PSA 2015 International Topical Meeting on Probabilistic Safety Assessment and Analysis, 2013.

19. “Pressurized Water Reactor Main Steam Line Break (MSLB) Benchmark,” Volume I: Final Specifications, NEA/NSC/DOC, Volume 99, Issue 8.