INL/EXT-15-35933
Revision 0

Light Water Reactor Sustainability Program

Improved Sampling Algorithms in the Risk-Informed Safety Margin Characterization Toolkit

August 2015

DOE Office of Nuclear Energy

INL/EXT-15-35933
Revision 0

Light Water Reactor Sustainability Program

Improved Sampling Algorithms in the Risk-Informed Safety Margin Characterization Toolkit

D. Mandelli, C. Smith, A. Alfonsi, C. Rabiti, J. Cogliati

August 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 RISMC approach is developing an advanced set of methodologies and algorithms in order to perform Probabilistic Risk Analyses (PRAs). In contract to classical PRA methods, which are based on Event-Tree and Fault-Tree methods, the RISMC approach largely employs system simulator codes applied to stochastic analysis tools. The fundamental approach via simulation is to randomly perturb (by employing sampling algorithms) timing and sequencing of events and uncertain parameters of the physics-based models (e.g., system codes such as RELAP-7) in order to estimate stochastic outcomes such as off-normal and damage states of the facility. Further, these outcomes can be used to estimate other useful metrics such as the plant core damage probability. This modeling approach applied to complex systems such as nuclear power plants requires the analyst to perform a series of computationally-expensive simulation runs given a large set of uncertain parameters. Consequently, these types of analysis are potentially affected by two issues. Firstly, the space of the possible solutions (the issue space or the response surface) can be sampled only sparsely and this precludes the ability to fully analyze the impact of uncertainties on the system dynamics. Secondly, large amounts of data are generated and tools to generate knowledge from such data sets have not typically been used for safety analysis approaches. This report focuses on the first issue and, in particular, describes how we can use novel methods that optimize the information generated by the sampling process by sampling unexplored or risk-significant regions of the issue space; we call this approach adaptive (smart) sampling algorithms. These methods infer the system response using surrogate models constructed from existing samples and predict the best location of the next sample. By using enhanced methods, it is possible to understand features of the issue space with a smaller number of carefully selected samples. In this report, we will present how it is possible to perform adaptive sampling using the RISMC toolkit and highlight the advantages compared to more classical sampling approaches such as Monte-Carlo. We will employ RAVEN to perform such statistical analyses applied to both analytical cases and more complex cases using RELAP-7.

CONTENTS

EXECUTIVE SUMMARY ii

FIGURES v

TABLES vi

ACRONYMS vii

1. Introduction 9

    1.1 Structure of the report 11

2. RISMC Approach 12

    2.1 RISMC toolkit 13

    2.2 RELAP-7 14

3. Sampling Methods 15

    3.1 Monte-Carlo 15

    3.2 LHS 16

    3.3 Grid 17

4. Adaptive Sampling 19

    4.1 Adaptive Sampling Algorithm 19

    4.2 Objective Functions: Limit Surface 20

5. Reduced Order Model (ROM) 22

    5.1 Examples of ROMs 23

        5.1.1 SVM 23
        5.1.2 Inverse-Weight Interpolator 24

    5.2 Convergence Criteria 25

6. RAVEN 26

    6.1 Simulation controller 26

    6.2 RAVEN Statistical Framework 27

    6.3 CROW 28

    6.4 RAVEN Input File Structure 29

7. Test cases 30

7.1 Analytical cases 30

7.1.1 Single region 30
7.1.2 Multiple regions 33
7.1.3 Convex 36

7.2 RELAP-7 Test Case 39

7.2.1 PWR SBO Limit surfaces 41
7.2.2 Limit surface for 100%-120% core power levels 43
7.2.3 PWR SBO probability calculation 44

8. Conclusions 46

References 48

FIGURES

Figure 1 – The approach used to support RISMC analysis 10

Figure 2 – Overview of the RISMC approach 12

Figure 3 – Overview of the RISMC toolkit 14

Figure 4 – Example of generation of 6 samples by using LHS sampling for a 2-dimensional case 17

Figure 5 – Grid sampling for a 2-dimensional case and probability weight calculation for a single cell 18

Figure 6 – Workflow of adaptive sampling algorithms 20

Figure 7 – Example of limit surface calculation for two different values of core power levels [30] 21

Figure 8 – Example of reduced order modeling approximation of a sampled 3-D response surface 22

Figure 9 – Limit surface evaluation using SVMs 24

Figure 10 – RAVEN simulation controller scheme 27

Figure 11 – Scheme of RAVEN statistical framework components 28

Figure 12 – Analytical shape of the single region limit surface 30

Figure 13 – Single region limit surface: sample locations and the estimated limit surface for different adaptive sampling iterations (1, 10, 20 and 40) 31

Figure 14 – Single region limit surface: sample locations and the estimated limit surface for different adaptive sampling iterations (60, 120 and 170) 32

Figure 15 – Analytical shape of the multiple regions limit surface 33

Figure 16 – Multiple regions limit surface: sample locations and the estimated limit surface for different adaptive sampling iterations (1, 30, 60 and 150) 34

Figure 17 – Multiple regions limit surface: sample locations and the estimated limit surface for different adaptive sampling iterations (250 and 371) 35

Figure 18 – Analytical shape of the convex limit surface 36

Figure 19 – Convex limit surface: sample locations and the estimated limit surface for different adaptive sampling iterations (1, 30, 60 and 150) 37

Figure 20 – Convex region limit surface: sample locations and the estimated limit surface for different adaptive sampling iterations (250, 350 and 456) 38

Figure 21 – Scheme of the TMI PWR benchmark 40

Figure 22 – Example of LOOP scenario followed by DGs failure to run using the RELAP-7 code 40

Figure 23 – RELAP-7 limit surface: sample locations and the estimated limit surface for different adaptive sampling iterations (1, 10, 30 and 60)..... 42

Figure 24 – RELAP-7 limit surface: sample locations and the estimated limit surface for different adaptive sampling iterations (100, 150 and 185)..... 43

Figure 25 – Limit surface obtained for two different levels of core power: 100% (left) and 120% (right) 44

Figure 26 – PWR SBO probability calculation for different pdfs of AC recovery time..... 45

Figure 27 – Original (left) and updated (right) pdfs for AC recovery time 45

Figure 28 – Multi-layer PRA 47

TABLES

Table 1 – Number of samples required to evaluate the single region limit surface by using classical Monte-Carlo and adaptive sampling (convergence in value is equal to 5 10-5)..... 33

Table 2 – Number of samples required to evaluate the multiple region limit surface by using classical Monte-Carlo and adaptive sampling (convergence in value is equal to 5 10-5)..... 35

Table 3 – Number of samples required to evaluate the convex limit surface by using classical Monte-Carlo and adaptive sampling (convergence in value is equal to 5 10-5) 38

Table 4 – Number of samples required to evaluate the RELAP-7 PWR SBO limit surface by using classical Monte-Carlo and adaptive sampling (convergence in value is equal to 5 10-5) 41

Table 5 – Conditional Core Damage Probability (CCDP) for two difference values of pdfs associated to AC recovery time: without (Original) and with (Updated) FLEX-like system..... 45

ACRONYMS

BWR	Boiling Water Reactor
CDF	Cumulative Distribution Function
CCD	Condition Core Damage
DOE	Department of Energy
DG	Diesel generator
EOP	Emergency Operating Procedures
ET	Event-Tree
FT	Fault-Tree
FW	Firewater
GPM	Gaussian Process Model
GUI	Graphical User Interface
IE	Initiating Event
INL	Idaho National Laboratory
LHS	Latin Hypercube Sampling
LOOP	Loss Of Offsite Power
LWR	Light Water Reactor
LWRS	Light Water Reactor Sustainability
MC	Monte-Carlo
MOOSE	Multi-physics Object-Oriented Simulation Environment
NPP	Nuclear Power Plant
PDF	Probability Distribution Function
PRA	Probabilistic Risk Assessment
PWR	Pressurized Water Reactor
R&D	Research and Development

RISMC	Risk Informed Safety Margin Characterization
ROM	Reduced Order Model
RPV	Reactor Pressure Vessel
SAMG	Severe Accident Management Guideline
T-H	Thermal-Hydraulics

Improved Sampling Algorithms in the Risk-Informed Safety Margin Characterization

1. Introduction

In the Risk Informed Safety Margin Characterization (RISMC) [1] approach, what we want to understand is not just the frequency of an event like core damage, but how close we are (or not) to key safety-related events and how might we increase the *safety margin*. A *safety margin* can be characterized in one of two ways:

- A deterministic margin, typically defined by the ratio (or, alternatively, the difference) of a capacity (i.e., strength) over the load
- A probabilistic margin, defined by the probability that the load exceeds the capacity. A probabilistic safety margin is a numerical value quantifying the probability that a safety metric (e.g., for an important process observable such as clad temperature) will be exceeded under accident scenario conditions.

The RISMC Pathway uses the probabilistic methods to determine safety margins and quantify their impacts to reliability and safety for existing Nuclear Power Plants (NPPs), i.e., pressurized and boiling water reactors (PWRs and BWRs). As part of the quantification, we use both probabilistic (via risk simulation) and mechanistic (via system simulators) approaches, as represented in Figure 1. Probabilistic analysis is represented by the risk analysis while mechanistic analysis is represented by the plant physics calculations. In the plant simulation, all the deterministic aspects that characterize system dynamics (e.g., thermo-hydraulic, thermo-mechanics, neutronics) are coupled to each other.

The risk simulation contains all deterministic elements that impact accident evolution from a controller point of view such as:

- Safety systems control logic
- Accident scenario initial and boundary conditions

in addition to stochastic ones such as:

- System/components failures
- Stochastic perturbation of internal elements of the physics simulation

The stochastic analysis [2] is performed in two steps:

1. sampling the stochastic parameters, and
2. evaluating the system response for the given set of sampled parameters

Step 1 often requires a large amount of samples; this is due to the fact that the number of stochastic parameters is very large and the uncertainties may be significant. The scope of the analysis is not only to determine outcome variables such as core damage probability but, more in general, evaluate the system response for different combinations of the stochastic parameters in order to determine specific observable outcomes (e.g., plant damage). In addition, a single evaluation of the system response in Step 2 is often computationally expensive. This is often the case when multi-physics simulator codes and/or multiple codes are employed in an integrated simulation run.

These two issues, large number of samples required and computational cost of each simulation run, make it challenging to perform a full PRA analysis of complex systems such as nuclear power plants. In order to overcome this challenge, two possible solutions are available [3]:

- Solution 1: Reduce the required number of samples
- Solution 2: Reduce the computational cost of each simulation run

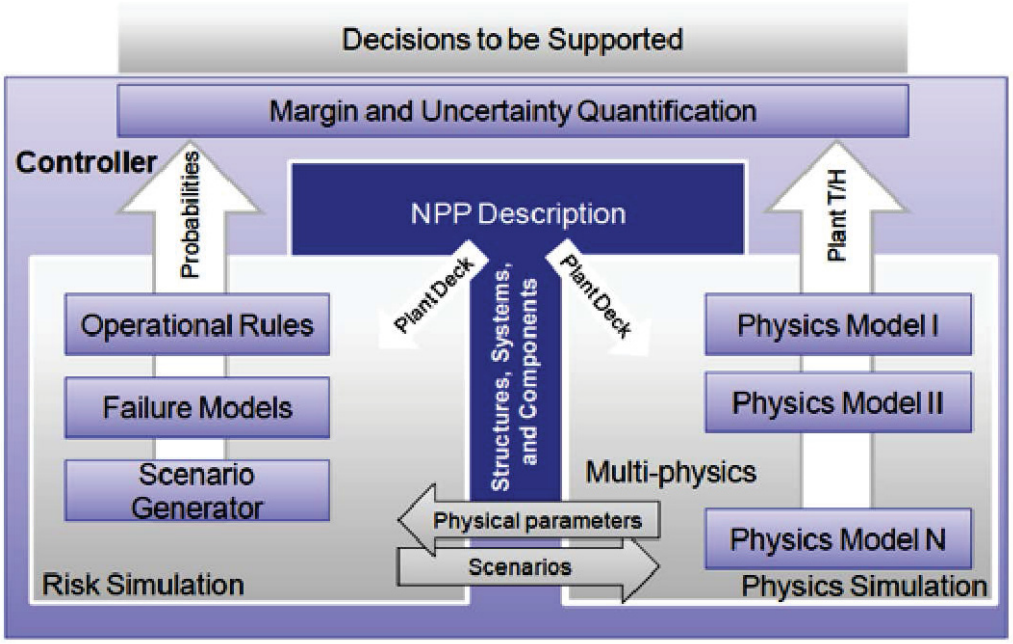

Figure 1 – The approach used to support RISMC analysis

Both solutions rely on the development of Reduced Order Models (ROMs) [4], also known as surrogate models. ROMs are mathematical models that infer the structure of a given set of data points using a blend of regression and interpolation techniques. In Solution 1, ROMs are built in order to generate the minimum set of samples that allows to determine the outcome variables: this is called *smart or adaptive sampling*. On the other hand, Solution 2 aims to build a ROM as a substitute of the simulation code itself (i.e., a surrogate for a complex calculation). ROMs are in fact much quicker to run (by factors of 100s to 1,000s) but, ROMs only approximate the simulation code and thus an intrinsic uncertainty is always present.

This report focuses on Solution 1. We will describe the logic behind the process of choosing the best set of samples that allows software to determine the desired outcome variables (e.g., core damage probability).

1.1 Structure of the report

This report is structured as follows:

- Section 2 gives a detailed overview of the RISMC approach and the RISMC toolkit
- Section 3 gives an overview of the classical sampling strategies
- Section 4 introduces the concept of adaptive sampling and how such methodology is implemented
- Section 5 describes the concept of ROM
- Section 6 shows the RAVEN statistical framework
- Section 7 presents a series of test cases to prove the validity of adaptive sampling compared to classical sampling methodologies

2. RISMC Approach

The RISMC approach employs both deterministic and stochastic methods in a single analysis framework (see Figure 2). In the deterministic method set we include:

- Modeling of thermal-hydraulic behavior of plant [5]
- Modeling of external events such as flooding [6]
- Modeling of the operators' responses to the accident scenario [7]

Note that deterministic modeling of plant or external events can be performed by employing specific simulator codes but also surrogate models (see Section 5), known as reduced order models (ROM). ROMs would be employed in order to decrease the high computational costs of employed codes.

In addition, multi-fidelity codes can be employed to model the same system; the idea is to switch from low-fidelity to high-fidelity code when higher accuracy is needed (e.g., use low-fidelity codes for steady-state conditions and high-fidelity code for transient conditions)

On the other hand, in the stochastic modeling we include all stochastic parameters that are of interest in the PRA analysis such as:

- Uncertain parameters
- Stochastic failure of system/components

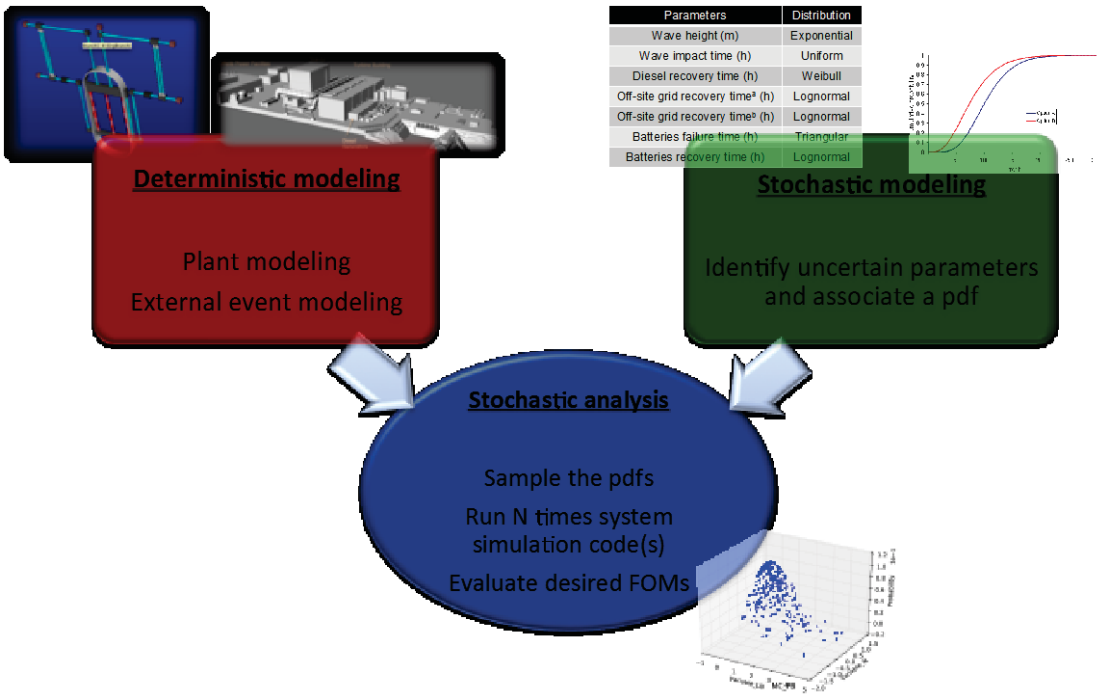

Figure 2 – Overview of the RISMC approach

As mentioned earlier, the RISMC approach heavily relies on multi-physics system simulator codes (e.g., RELAP [5]) coupled with stochastic analysis tools (e.g., RAVEN [8]). From a mathematical point of view, a single simulator run can be represented as a single trajectory in the phase space. The evolution of such a trajectory in the phase space can be described as follows:

$$\frac{\partial \boldsymbol{\theta}(t)}{\partial t} = \boldsymbol{\mathcal{H}}(\boldsymbol{\theta}, \boldsymbol{s}, t) \tag{1}$$

where:

- $\boldsymbol{\theta} = \boldsymbol{\theta}(t)$ represents the temporal evolution of a simulated accident scenario, i.e., $\boldsymbol{\theta}(t)$ represents a single simulation run
- $\boldsymbol{\mathcal{H}}$ is the actual simulator code that describes how $\boldsymbol{\theta}$ evolves in time
- $\boldsymbol{s} = \boldsymbol{s}(t)$ represents the status of components and systems of the simulator (e.g., status of emergency core cooling system, AC system)

By using the RISMC approach, the PRA analysis is performed by [2]:

1. Associating a probabilistic distribution function (pdf) to the set of parameters $\boldsymbol{s}$ (e.g., timing of events)
2. Performing stochastic sampling of the pdfs defined in Step 1
3. Performing a simulation run given $\boldsymbol{s}$ sampled in Step 2, i.e., solve Eq. (1)
4. Repeating Steps 2 and 3 *M* times and evaluating user defined stochastic parameters such as CD probability (P_{CD}).

2.1 RISMC toolkit

In order to perform advanced safety analysis, the RISMC Pathway has a toolkit that was developed at INL using MOOSE [9] as the underlying numerical solver framework. This toolkit consists of the following software tools (see Figure 3):

- RELAP-7 [5] (see Section 2.2): the code responsible for simulating the thermal-hydraulic dynamics of the plant.
- RAVEN [8] (see Section 6): it has two main functions: 1) act as a controller of the RELAP-7 simulation and 2) generate multiple scenarios (i.e., a sampler) by stochastically changing the order and/or timing of events.
- PEACOCK [10] (see Section 2.3): the Graphical User Interface (GUI) that allows the user to create/modify input files of both RAVEN and RELAP-7 and it monitors the simulation in real time while it is running.

- GRIZZLY [11]: the code that simulates the thermal-mechanical behavior of components in order to model component aging and degradation. Note that for the analysis described in this report, aging was not considered.

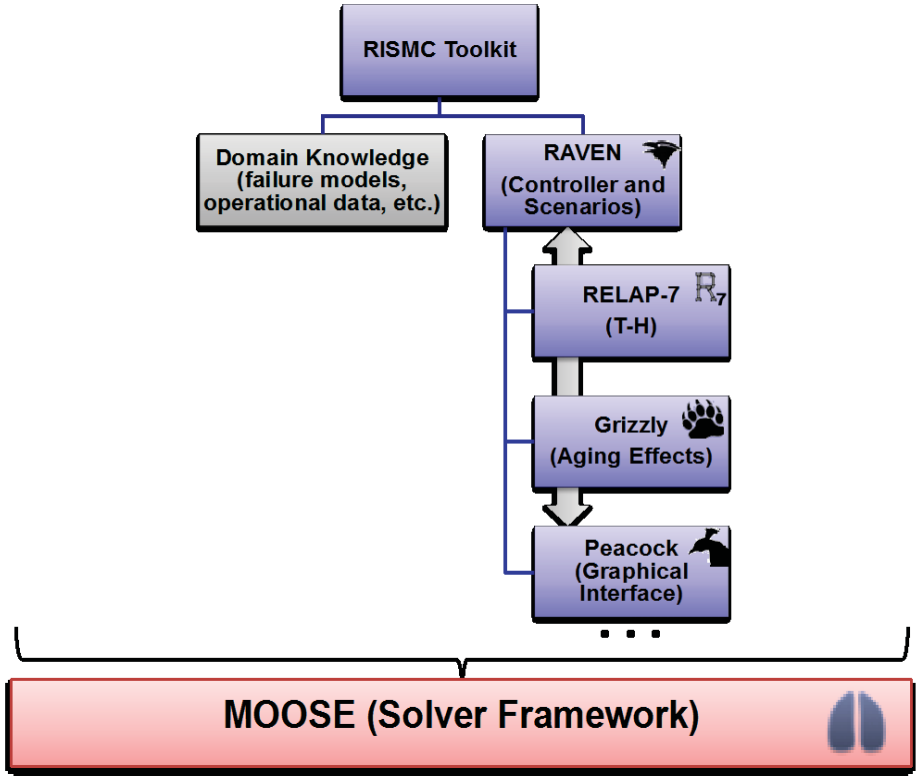

Figure 3 – Overview of the RISMC toolkit

2.2 RELAP-7

The RELAP-7 code [5] is the new nuclear reactor system safety analysis code being developed at the Idaho National Laboratory (INL). RELAP-7 is designed to be the main reactor system simulation toolkit for the RISMC Pathway. The RELAP-7 code development is taking advantage of the progress made in the past several decades to achieve simultaneous advancement of physical models, numerical methods, and software design. RELAP-7 uses the INL's MOOSE (Multi-Physics Object-Oriented Simulation Environment) framework [8] for solving computational engineering problems in a well-planned, managed, and coordinated way. This allows RELAP-7 development to focus strictly on systems analysis-type physical modeling and gives priority to retention and extension of RELAP5's multidimensional system capabilities.

A real reactor system is complex and may contain hundreds of different physical components. Therefore, it is impractical to preserve actual geometry for the entire system. Instead, simplified thermal-hydraulic models are used to represent (via "nodalization") the major physical components and describe major physical processes (such as fluid flow and heat transfer). There are three main types of components developed in RELAP-7: (1) one-dimensional (1-D) components, (2) zero-dimensional (0-D) components for setting a boundary, and (3) 0-D components for connecting 1-D components.

3. Sampling Methods

This section summarizes the three most used sampling strategies that are often employed for both PRA and UQ/SA applications. These are:

- Monte-Carlo [12] (see Section 3.1)
- Latin-hypercube sampling (LHS) [13] (see Section 3.2)
- Grid (see Section 3.3)

In the next sections, these sampling methods are described, and for all these methods we assume the following:

1. A fixed set of stochastic parameters have been chosen
2. A unique probability distribution function is associated to each stochastic parameter. For the scope of this report we will not cover issues related to coupled stochastic variables
3. The outcome (either Boolean or continuous) of the simulator is always repeatable, i.e., the simulator does not contain intrinsic stochastic elements[1]

3.1 Monte-Carlo

Monte-Carlo [12] method is the most used algorithm for simulation-based reliability calculations. For a more detailed presentation of the methodology we use [14] as a reference which we have summarized here very briefly. The object of the Monte-Carlo methodology is to numerically evaluate the expected value $E[.]$ of a generic function $g(\omega)$ over a specified domain D in the space. In the most generic case the expected value of $g(\omega)$ is weighted by a function $f(\omega)$ which may vary within D as shown below:

$$E[g(\omega)] = \int_D g(\omega)\, f(\omega)\, d\omega = G \qquad \text{Eq. 1}$$

As part of the analysis we are interested in evaluation of the second order moment of $g(\omega)$ (the first order moment of $g(\omega)$ is $E[g(\omega)]$), i.e. its variance $Var[g(\omega)]$:

[1] For our applications we will consider simulators that numerically solve systems of either ordinary or partially differential equations. This hypothesis is valid for most cases such as for the RELAP-7 case. This condition does not apply if stochastic models (e.g., Markov Models or MM1 queue models) are embedded in the simulator itself.

$$Var[g(\omega)] = \int_D (g(\omega) - G)^2 \, pdf(\omega) \, d\omega = E[g^2(\omega)] - G^2 \quad \text{Eq. 2}$$

In our applications, the weighting function has a specific meaning: $f(\omega)d\omega$ is the probability associated to the element $d\omega$. So, Eq. 1 can now be rewritten as:

$$G = \int_D g(\omega) \, pdf(\omega) d\omega \quad \text{Eq. 3}$$

The basic idea of the Monte-Carlo methodology is to evaluate the integral shown above by:

1. Sampling the input space, i.e., the space of the stochastic parameters by creating a series a samples coordinates ω_i
2. Collecting the outcome $g(\omega_i)$
3. Evaluate the first and second statistical moments:

$$E[g(\omega_i)] = \frac{1}{N} \sum_{i=1}^{N} g(\omega_i) \quad \text{Eq. 4}$$

$$Var[g(\omega_i)] = \frac{\frac{1}{N} \sum_{i=1}^{N} g^2(\omega_i) - \left(\frac{1}{N} \sum_{i=1}^{N} g(\omega_i) \right)^2}{N - 1} \quad \text{Eq. 5}$$

It can be shown that for a large number of samples, i.e. $N \gg 1$ we can approximate:

$$E[g(\omega)] \approx \frac{1}{N} \sum_{i=1}^{N} g(\omega_i) \quad , \quad E[g^2(\omega)] \approx \frac{1}{N} \sum_{i=1}^{N} g^2(\omega_i) \quad \text{Eq. 6}$$

3.2 LHS

Latin hypercube sampling method [13] is another widely used a statistical method for generating a series of samples from a set of N distributions. Statistical theory behind LHS is very similar to the one presented in Section 3.1; the major difference is how the samples are created. Samples are in fact not created by performing the inverse of the cdf of each stochastic parameter. In the general case when M samples are required, the range of each variable is divided into M equally probable intervals, thus creating a Cartesian grid in the variable space. Each of the M samples are chosen such that each discretization of each variable contains only one sample (i.e., it follows the requirements of a Latin square).

Figure 4 gives an example of LHS sampling for the generation of six samples from a set of two independent stochastic variables (i.e., $M = 6$, $N = 2$). A Latin square composed by 36 cells in a 6×6 Cartesian grid. The coordinate of each sample in each cell is uniformly sampled within the cell itself.

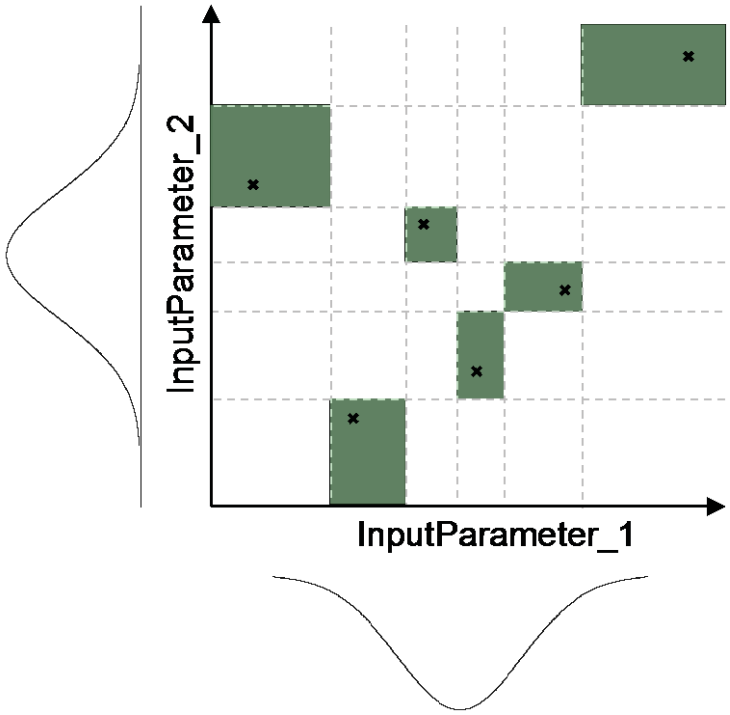

Figure 4 – Example of generation of 6 samples by using LHS sampling for a 2-dimensional case

3.3 Grid

Probably the simplest sampling strategy, the grid sampler simply builds a Cartesian grid in the input space (see Figure 5). Samples point coordinates are the centers of all cells of the grid. The total number of samples is equal to the product of the discretization of every dimension. Thus, for analyses that involve many stochastic parameters (i.e., many dimensions), the number of required samples is very large.

For each sample, a probability weight w_{pb} is associated to it:

$$w_{pb} = \int_{cell} pdf(\omega)\, d\omega \qquad \text{Eq. 7}$$

If the input parameters are independent, then $pdf(\omega)$ is the product of the pdfs of each input parameter.

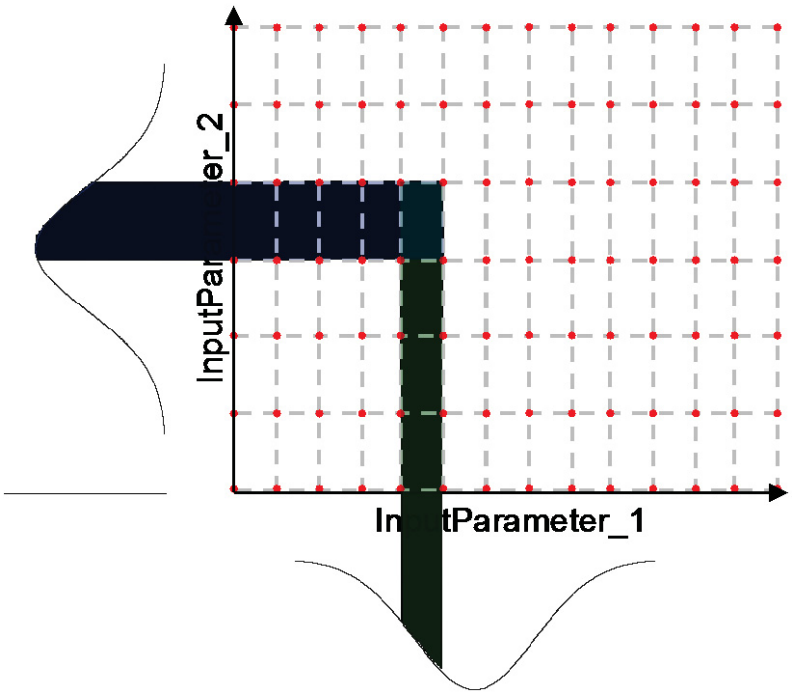

Figure 5 – Grid sampling for a 2-dimensional case and probability weight calculation for a single cell

4. Adaptive Sampling

The Sampling methods presented in Section 3 have the disadvantage that they require a large number of samples. This can be unfeasible when each sample may require large computational resources. This is especially true when large thermal-hydraulic codes such as RELAP-7 are used.

In order to overcome these limitations, it is possible to employ improved sampling strategies, i.e., adaptive sampling algorithms. The basic idea is to combine sampling process with ROMs. ROMs are employed to:

- Infer the response of the system
- Predict where is the location of the samples that maximize the information from the calculation

In this section we will describe in detail what are the basic steps behind adaptive sampling algorithms.

4.1 Adaptive Sampling Algorithm

The general adaptive sampling pipeline begins by selecting some initial training data, running the simulation and obtaining a collection of true responses at these data points. Second, it fits a response surface surrogate model from the initial set of training data. Third, a set of candidate points is chosen in the parameter space based on certain sampling technique, and the surrogate model is evaluated at these points, obtaining a set of approximated values. Fourth, each candidate point is assigned a score based on some adaptive sampling scoring function (usually derived from qualitative or quantitative relations between the training points, their true and estimated response values). Finally, the candidate(s) with the highest score(s) are selected and added to the set of training data to begin a new cycle.

As mentioned earlier this kind of sampling strategy requires not only simulator codes but also one, or possibly more, ROMs [4]. In our case, it is possible to view the code as a black-box $\boldsymbol{\mathcal{H}}$ that produces a set of output variables $\boldsymbol{y}$ given a set of input parameters $\boldsymbol{s}$:

$$\boldsymbol{\mathcal{H}}: \boldsymbol{s} \rightarrow \boldsymbol{y}(t) = \boldsymbol{\mathcal{H}}(\boldsymbol{\theta}, \boldsymbol{s}, t) \qquad \text{Eq. 8}$$

In addition, it is needed to provide what we call an "objective function". The objective function gives indications on what is the desired "exploration" criteria. We will give more description of the objective functions in Section 4.2.

The main adaptive sampling steps are explained as follows (see Figure 6):

1. Perform a set of runs the simulator code: the number of required runs may depend on the dimensionality of the input space.
2. Given the set of simulation runs obtained in Step 1, create a ROM. The objective of this ROM is to:
    - Infer the response of the simulator code, i.e., create an approximate output given the same set of input parameters

- Predict region in the input space that maximize the objective function

3. Employ the ROM to approximate the structure of the goal function
4. Identify a set of points that satisfy the conditions specified in the goal function
5. Chose a subset of points from the ones obtained in Step 4 that maximize the goal function
6. Perform a simulation run for each of the points obtained in Step 5 using the simulator code
7. Repeat Steps 2 through 6 until convergence is reached

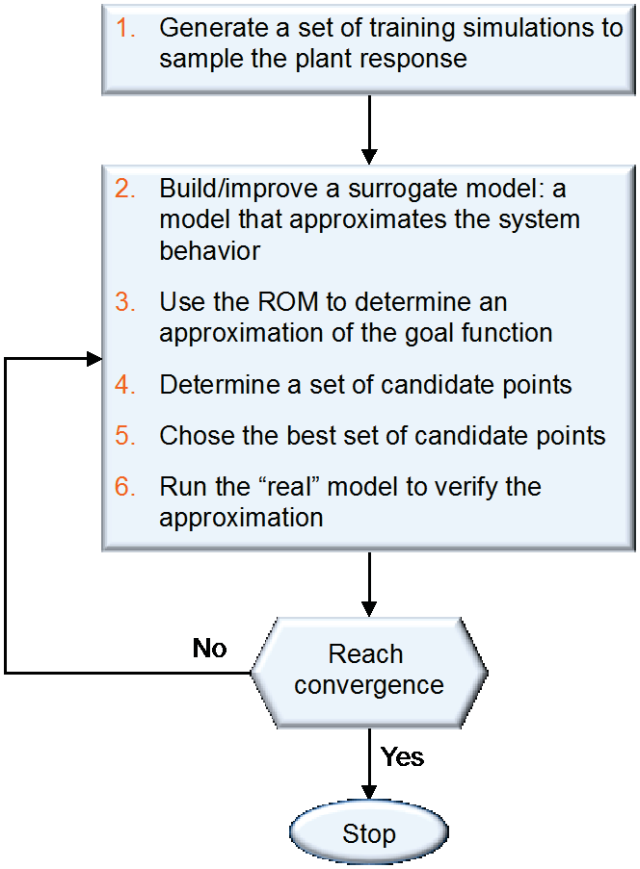

Figure 6 – Workflow of adaptive sampling algorithms

4.2 Objective Functions: Limit Surface

As part of safety margin quantification, the RISMC approach aims to evaluate a set of limit surfaces [16]. A limit surface represents the boundaries in the input space (i.e., d-dimensional space; each dimension is one the d sampled variables) that separate failure region (i.e., characterized by the undesired

simulation outcome; e.g., core damage) from success region (i.e., characterized by the desired simulation outcome; e.g., max clad temperature below 2200 F).

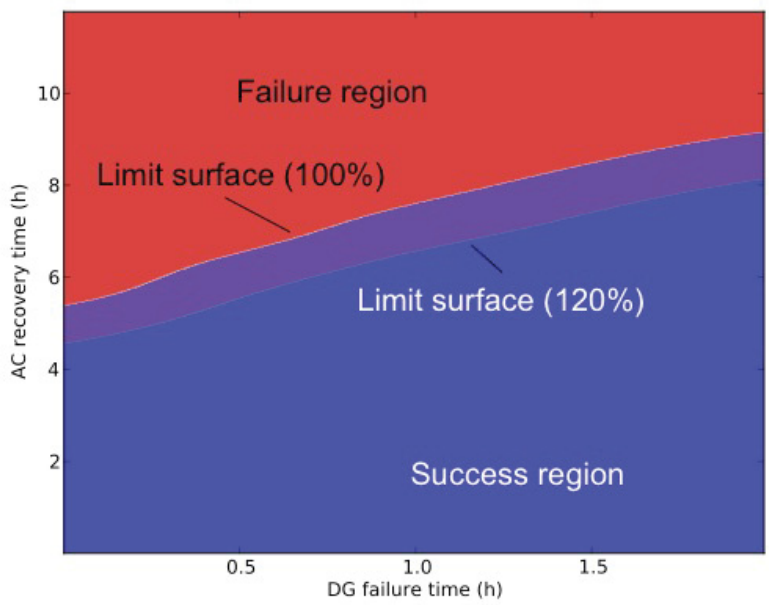

Figure 7 – Example of limit surface calculation for two different values of core power levels [30]

The limit surface has a pure deterministic value; the stochastic information is generated when the probability of occurrence of the undesired event (e.g., core damage) P_{CD} is determined as:

$$P_{CD} = \int_{\substack{failure \\ region}} pdf(\varpi)\, d\varpi \qquad \text{Eq. 9}$$

Equation 9 shows that P_{CD} is equal to the area of the failure region weighted by the probability of being in the failure region itself (through the probability distribution function $pdf(\varpi)$).

Figure 7 shows the limit surface in a 2-dimensional space generated in [17] using RAVEN coupled with RELAP-7 for a boiling water reactor station blackout initiating event. As part of the analysis, we were interested in the evaluation of the safety impacts of power uprate (reactor core power increased from 100 to 120%). Such evaluation has been performed by evaluating both the increased core damage probability ΔP_{CD} and the limit surface for both 100 and 120% reactor core power level

Note that ΔP_{CD} can be written as the same integral indicated in Eq. 9 but evaluated only in the *expanded failure region* ($\Delta\Omega_{Failure}$)

$$\Delta P_{CD} = \int_{\Delta\Omega_{Failure}} pdf(\varpi)\, d\varpi \qquad \text{Eq. 10}$$

5. Reduced Order Model (ROM)

A ROM is a mathematical model that aims to build a correlation given a set of data points. The starting point is typically a set of *N* data points:

$$(\boldsymbol{s_i}, \boldsymbol{\mathcal{H}}(\boldsymbol{s_i})) \quad i = 1, \dots, N \qquad \text{Eq. 11}$$

that samples the response of the original model. Given the set of these *N* data points, the ROM is trained and the resulting outcome is a model $\boldsymbol{\Theta}(\boldsymbol{s})$ that approximates the original model $\boldsymbol{\mathcal{H}}(\boldsymbol{s})$(see Figure 2):

$$\boldsymbol{\Theta}(\boldsymbol{s}): \boldsymbol{s_i} \rightarrow \boldsymbol{\Theta}(\boldsymbol{s_i}) \cong \boldsymbol{\mathcal{H}}(\boldsymbol{s_i}) \qquad \text{Eq. 12}$$

The advantage of the ROM is the much faster computation of $\boldsymbol{\Theta}(\boldsymbol{s})$ (e.g., RELAP) compared to the original model $\boldsymbol{\mathcal{H}}(\boldsymbol{s})$. However, the evaluation of a ROM is affected by an intrinsic error which can not be bound and/or quantified.

We have identified two classes of ROM: model based and data based. These two classes are described in the next two sections.

In model based ROMs the prediction is performed using a blend of interpolation and regression algorithms. Examples are:

- Gaussian Process Models (GPMs) [18]
- Multi-dimensional spline interpolators [19]

This class of algorithms has the advantage that they possess great prediction capabilities if the original $\boldsymbol{\mathcal{H}}(\boldsymbol{s})$ is relatively smooth (i.e., not discontinuous).

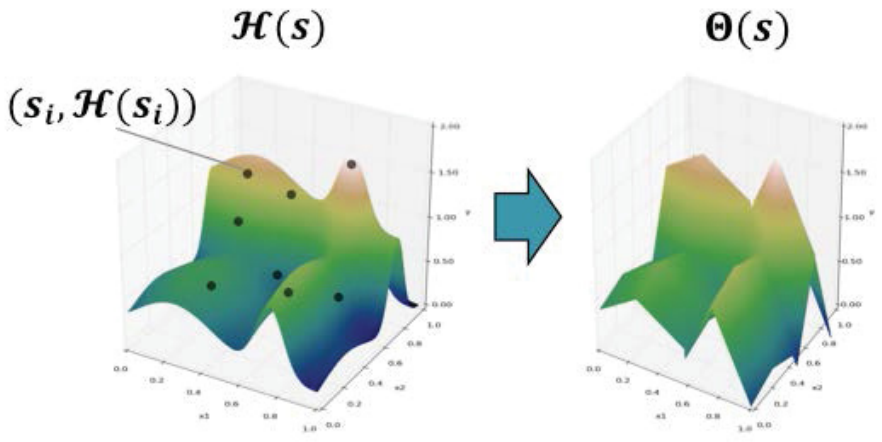

Figure 8 – Example of reduced order modeling approximation of a sampled 3-D response surface

In data based ROMs the prediction is performed by solely considering the input data by using data searching algorithms. Examples are:

- K Nearest Neighbor classifier (KNN) [20]
- Graph based models [21]

While the predictions of this class of ROMs is limited compared to model based ROMs, they have the advantage that they are able to handle very discontinuous $\boldsymbol{\mathcal{H}}(\boldsymbol{s})$.

5.1 Examples of ROMs

In this section we briefly describe two ROM that will be used in the test cases shown in Section 7. Even though RAVEN has a much larger variety of ROMs, it is not in the scope of this report to give a detailed description of the ROMs available in RAVEN or to perform a comparison among them. In this report we will limit ourselves to two ROMs: Support Vector Machines (SVM) [22, 23] (see Section 5.1.1) and the Inverse-Weight interpolator [43] (see Section 5.1.2).

5.1.1 SVM

This section explains how the limit surfaces shown in Section 5.3.2 have been evaluated. We employed Support Vector Machine [22, 23] based algorithms.

Given a set of *N* multi-dimensional samples $\boldsymbol{x}_i$ and their associated results $y_i = \pm 1$ (e.g., $y_i = +1$ for system success and $y_i = -1$ for system failure), the SVM finds the boundary (i.e., the decision function) that separates the set of points having different y_i. The decision function lies between the support hyper-planes, which are required to:

- Pass through at least one sample of each class (called support vectors)
- Not contain samples within them

For the linear case, see Figure 9, the decision function is chosen such that distance between the support hyper-planes is maximized.

Without going into the mathematical details, the determination of the hyper-planes is performed recursively and updated every time a new sample has been generated. Figure 9 shows the SVM decision function and the hyper-planes for a set of points in a 2-dimensional space having two different outcomes: $y_i = +1$ (green) and $y_i = -1$ (red).

The transition from a linear to a generic non-linear hyper-plane is performed using the kernel trick: i.e., by projecting of the original samples into a higher dimensional space known as featured space generated by kernel functions $K(\boldsymbol{x_i}, \boldsymbol{x_j})$:

$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \exp\left(-\frac{\|\boldsymbol{x_i} - \boldsymbol{x_j}\|}{2\sigma^2}\right) \quad \text{Eq. 13}$$

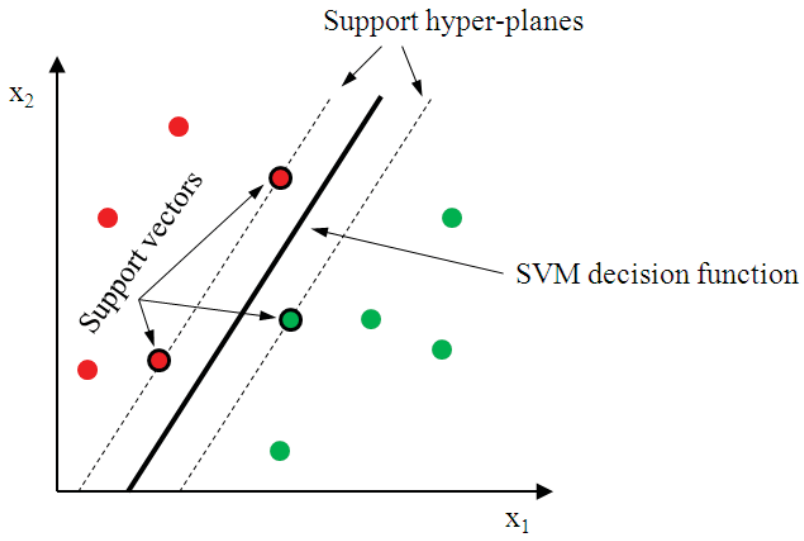

Figure 9 – Limit surface evaluation using SVMs

5.1.2 Inverse-Weight Interpolator

The Inverse-Weight interpolator [24], also know as Sheppard interpolator, is ROM completely different from SVM. It performs predictions not on binary but on continuous outcomes. The staring point is a set of N data points $(\boldsymbol{x}_i, u_i)\ i = 1, \dots, N$ where $\boldsymbol{x}_i$ are the coordinate in the input space $D \subset \mathbb{R}^n$ and u_i is the outcome. The Inverse-Weight interpolator can be represented as a function $u(\boldsymbol{x})$ that, given a coordinate in the input space $\boldsymbol{x}$, generates a prediction on u:

$$u(\boldsymbol{x}): \boldsymbol{x} \to \mathbb{R}\,, \quad \boldsymbol{x} \in D \subset \mathbb{R}^n \qquad \text{Eq. 14}$$

The prediction $u(\boldsymbol{x})$ is performed by summing all data points $u_i\ i = 1, \dots, N$ weighted by a weighting parameter $w_i(\boldsymbol{x})$ as follows:

$$u(\boldsymbol{x}) = \begin{cases} \dfrac{\sum_{i=1}^{N} w_i(\boldsymbol{x})\, u_i}{\sum_{i=1}^{N} w_i(\boldsymbol{x})} & \text{if } d(\boldsymbol{x}, \boldsymbol{x}_i) \neq 0 \\ u_i & \text{if } d(\boldsymbol{x}, \boldsymbol{x}_i) = 0 \end{cases} \qquad \text{Eq. 15}$$

where $w_i(\boldsymbol{x})$ is the inverse of the distance between $\boldsymbol{x}$ and $\boldsymbol{x}_i$:

$$w_i(\boldsymbol{x}) = \left(\frac{1}{d(\boldsymbol{x}, \boldsymbol{x}_i)}\right)^p \qquad \text{Eq. 16}$$

Large values of p assign greater weight $w_i(\boldsymbol{x})$ to data points $\boldsymbol{x}_i$ closest to $\boldsymbol{x}$, with the result turning into a mosaic of tiles (i.e., Voronoi diagram) with nearly constant interpolated value.

5.2 Convergence Criteria

During the adaptive sampling step, RAVEN continues to generate new sample coordinates (see Figure 6) until convergence is reached. RAVEN allows three types of convergence criteria:

1. Maximum number of samples

2. CDF-weight: the convergence is checked in terms of probability (Cumulative Distribution Function)

3. Value-weight: the convergence is checked on the hyper-volume in terms of variable values

In addition it is possible to specify a value of persistence which allows to create an additional convergence check. It represents the number of times the computed convergence error needs to be below the tolerance value given by the user before stopping the adaptive-sampling calculation.

6. RAVEN

RAVEN (Risk Analysis and Virtual control ENviroment) 8] is a software framework that acts as the control logic driver for the thermal-hydraulic code RELAP-7, a newly developed software at INL. RAVEN is also a multi-purpose Probabilistic Risk Assessment (PRA) code that allows for probabilistic analysis of complex systems. It is designed to derive and actuate the control logic required to simulate both plant control system and operator actions (guided procedures) and to perform both Monte-Carlo sampling [12] of random distributed events and dynamic branching-type [25] based analysis.

RAVEN consists of two main software components:

1. Simulation controller (see Section 6.1)
2. Statistical framework (see Section 6.2)

6.1 Simulation controller

One task of RAVEN is to act as controller of the RELAP-7 simulation while simulation is running. Such control action is performed by using two sets of variables [26]:

- *Monitored variables*: set of observable parameters that are calculated at each calculation step by RELAP-7 (e.g., average clad temperature)
- *Controlled parameters*: set of controllable parameters that can be changed/updated at the beginning of each calculation step (e.g., status of a valve – open or closed –, or pipe friction coefficient)

The manipulation of these two data sets of variables is performed by two components of the RAVEN simulation controller (see Figure 10):

- RAVEN control logic: is the actual system control logic of the simulation where, based on the status of the system (i.e., monitored variables), it updates the status/value of the controlled parameters
- RAVEN/RELAP-7 interface: is in charge of updating and retrieving RELAP-7/MOOSE component variables according to the control logic

A third set of variables, i.e. *auxiliary variables*, allows the user to define simulation specific variables that may be needed to control the simulation. From a mathematical point of view, auxiliary variables are the ones that guarantee the system to be Markovian [27], i.e., the system status at time $t = \bar{t} + \Delta t$ can be numerically solved given only the system status at time $t = \bar{t}$.

The set of auxiliary variables also includes those that monitor the status of specific control logic set of components (e.g., diesel generators, AC buses) and simplify the construction of the overall control logic scheme of RAVEN.

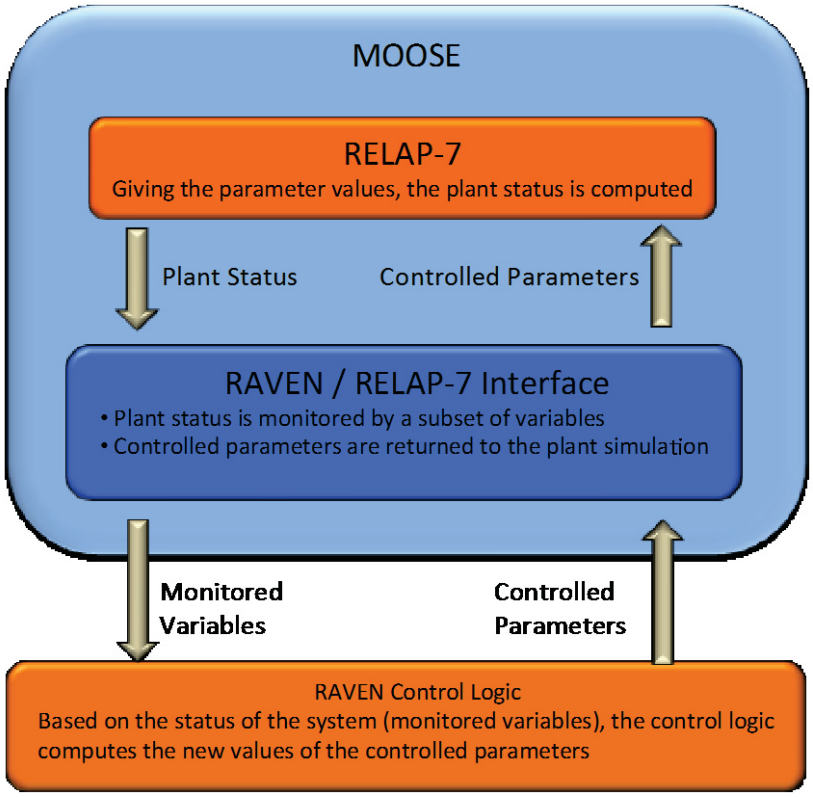

Figure 10 – RAVEN simulation controller scheme

6.2 RAVEN Statistical Framework

The RAVEN statistical framework is a recent add-on of the RAVEN package that allows the user to perform generic statistical analysis. By statistical analysis we include:

- Sampling of codes: either stochastic (e.g., Monte-Carlo [12] and Latin Hypercube Sampling [13]) or deterministic (e.g., grid and Dynamic Event Tree [25])
- Generation of Reduced Order Models [4] also known as Surrogate models
- Post-processing of the sampled data and generation of statistical parameters (e.g., mean, variance, covariance matrix)

Figure 4 shows a general overview of the elements that comprise the RAVEN statistical framework:

- Model: it represents the pipeline between input and output space. It comprises both codes (e.g., RELAP-7) and also Reduced Order Models
- Sampler: it is the driver for any specific sampling strategy (e.g., Monte-Carlo, LHS, DET)
- Database: the data storing entity

- Post-processing module: module that performs statistical analyses and visualizes results

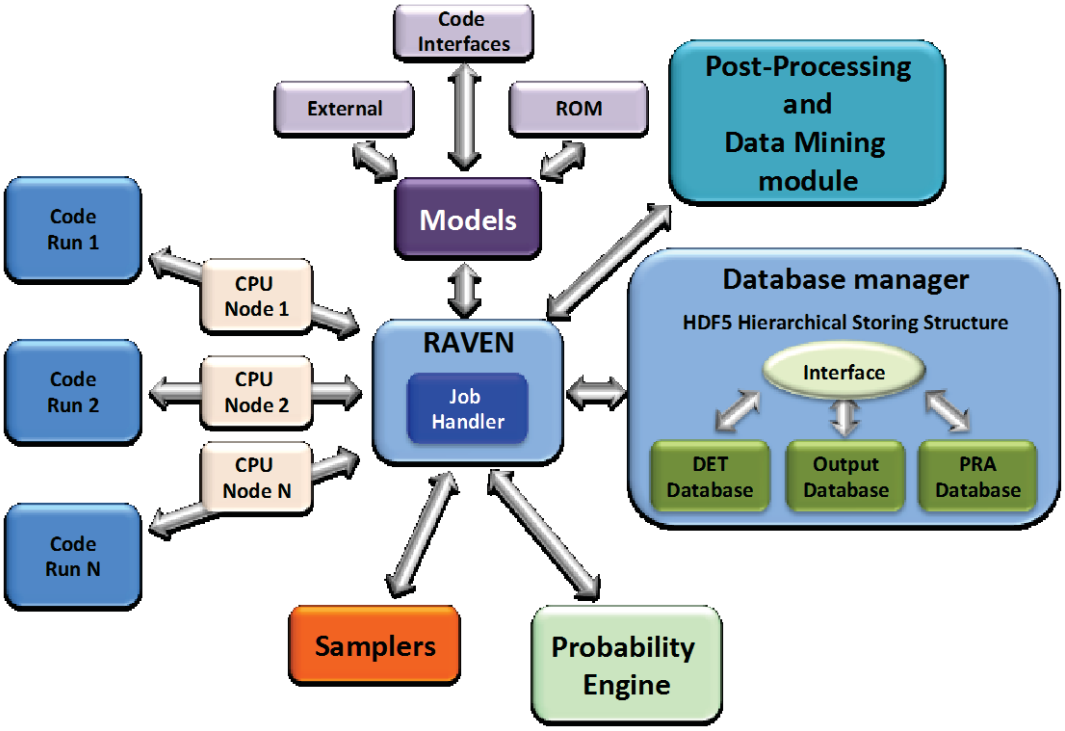

Figure 11 – Scheme of RAVEN statistical framework components

6.3 CROW

CROW is an INL internally developed C++ library which contains the full set of probabilistic functions which are used by RAVEN to perform any kind of statistic analysis. It contains the following modules:

- Interface with BOOST [28]. BOOST[2] is a set of libraries for the C++ programming language that provide support for tasks and structures such as linear algebra, pseudorandom number generation, multithreading, image processing, regular expressions, and unit testing. For our applications we are using the mathematical/statistical[3] library of BOOST which contains a wide selection of univariate statistical distributions and functions that operate on them (pdf and cdf calculation along with random number generation).

- Multi-variate distributions. This module contains the same functions mentioned above (pdf and cdf calculation along with random number generation) but applied to multi-variate distributions.

[2] http://www.boost.org/

[3] http://www.boost.org/doc/libs/1_58_0/libs/math/doc/html/dist.html

Not only multi-variate normal distributions are modeled but also generic multi-variate distributions defined over a set of sampled scattered or grid distributed.

6.4 RAVEN Input File Structure

All the information required by RAVEN to perform any kind of statistical analysis are specified in a single input file in .xml format. The .xml format allows users to build a high modular input file. A generic RAVEN input file contains a set of blocks as follows:

- **Entities**: these are main blocks that will be used
  - *DataObjects*: how data is stored within RAVEN
  - *Databases*: data storage entities
  - *Samplers*: input space sampling entities (e.g., MC, LHS)
  - *OutStreamManager*: used for data exporting/dumping (plotting and printing)
  - *Distributions*: probabilistic representation of variables
  - *Models*: projection from input to output space
  - *Functions*: user-defined functions
- **Steps**: each step links entities together to perform an action. Note that multiple heterogeneous Entities are used in a single Step (e.g., DataObjects + Samplers + Models). In each step, every entity has a role specified by the user (Input, Output, Model, Sampler, Function, ROM, Solution export). In addition, five types of steps are available:
  - *SingleRun*: perform a single run of a model
  - *MultiRun*: perform multiple runs of a model
  - *RomTrainer*: perform the training of a Reduced Order Model (ROM)
  - *PostProcess*: post-process data or manipulate RAVEN entities
  - *IOStep*: data management
- **RunInfo**: in this block the user specifies:
  - Sequence of steps to be performed
  - External files required
  - Specific computational parameters needed

7. Test cases

In this section we show how the methods and algorithms presented in the past sections are applied for particular sampling cases. These sampling cases cover both analytical problems (see Section 7.1) but also safety related applications using safety analysis codes such as RELAP-7 [5] (see Section 7.2). For these cases we employed SVMs with Gauss kernel (also know as radial basis function) as ROM to guide the choice of the adaptive sampling samples.

7.1 Analytical cases

This section shows the capabilities of adaptive sampling algorithms to sample specific regions of the input space. The starting point is a single region as described in Section 7.1.1. We then move to sampling a closed region (see Section 7.1.2) and multiple closed regions (see Section 7.1.3)

7.1.1 Single region

In this simple case, the limit surface is a single region located at the lower left corner of a 2-dimensional space. This is the most basic case that can be encountered. The model considered for this test case, given x_1 and x_2 produce an output variable y as:

$$y = x_1^2 + x_2 - 0.5 \qquad \text{Eq. 17}$$

Figure 12 shows the surface representing y. In addition, this test case failure occurs when $y > 0$ (red plane in Figure 12). The analytical limit surface is the red line shown in Figure 12.

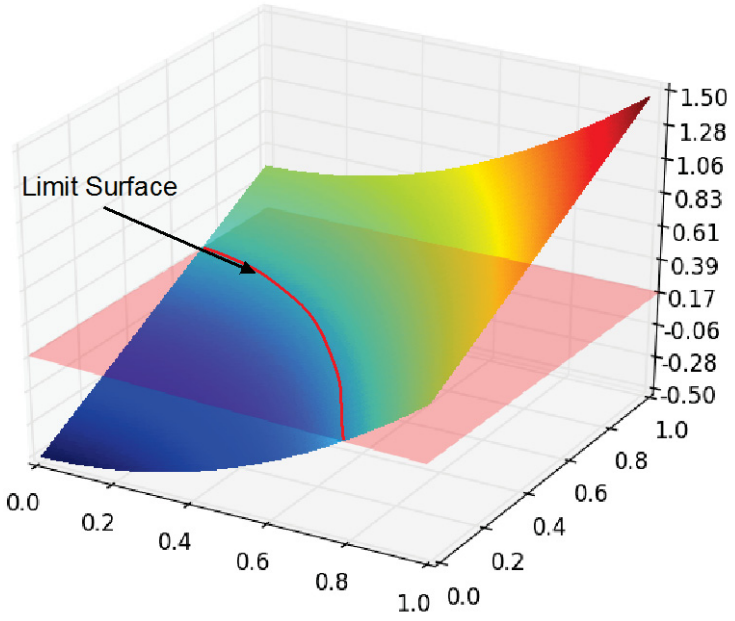

Figure 12 – Analytical shape of the single region limit surface

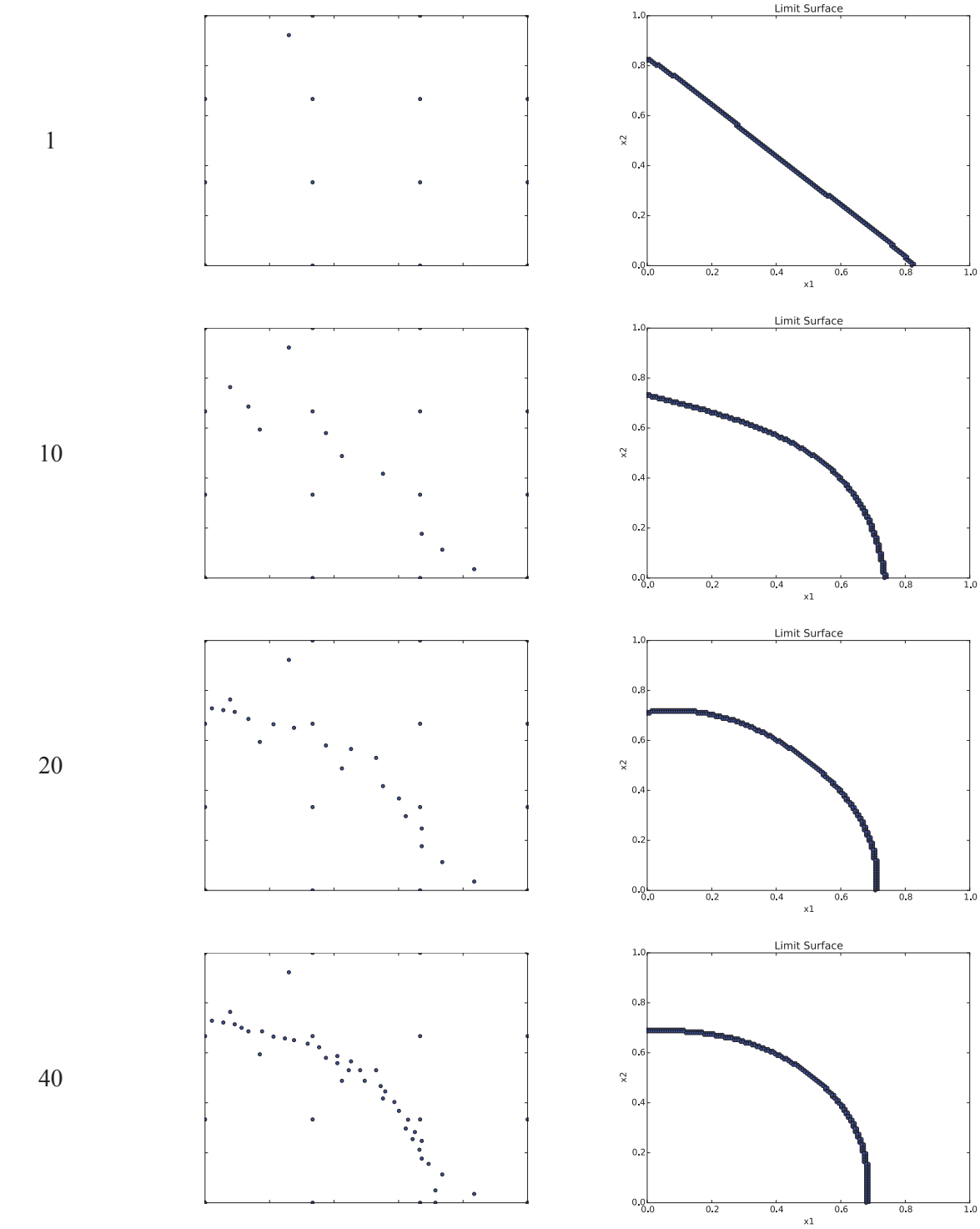

Figure 13 – Single region limit surface: sample locations and the estimated limit surface for different adaptive sampling iterations (1, 10, 20 and 40)

The scope of this test is to show explicitly how the adaptive sampling process determines the most likely location of the limit surface and the choice of the next sample. We performed the search of this limit using the adaptive sampling capabilities available within RAVEN. The analytical formulation of the response surface of the system has been implemented using a python script interfaced directly with RAVEN (as an external model).

Figure 13 and Figure 14 show a summary of the adaptive sampling process generated using RAVEN. Each row in the plots of Figure 13 and Figure 14 corresponds to an iteration of the adaptive sampling process. For each iteration two plots are shown: sample locations (left) and the estimated limit surface (right). For this case convergence (convergence in value is equal to $5\ 10^5$) was reached after 170 samples.

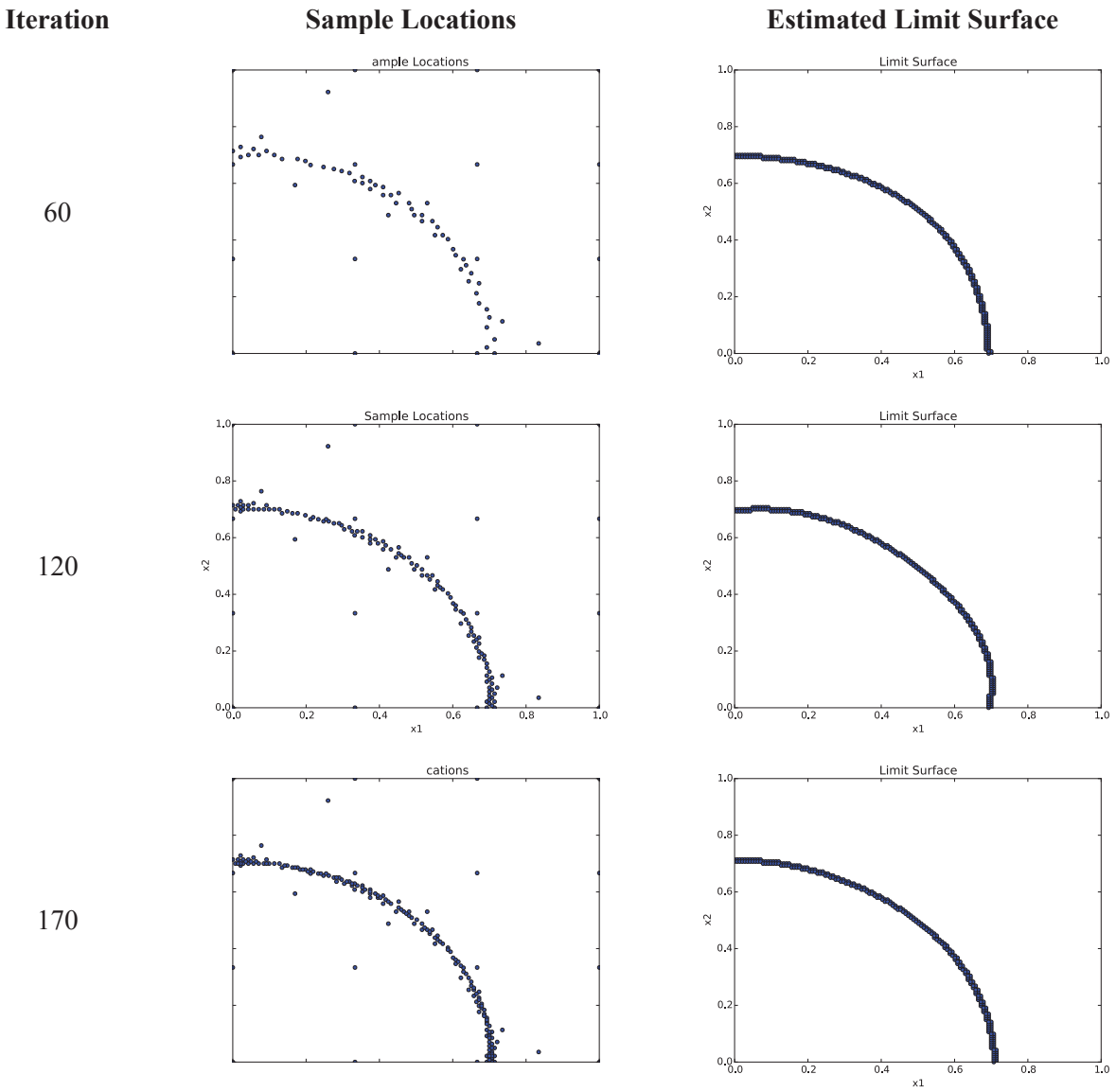

Figure 14 – Single region limit surface: sample locations and the estimated limit surface for different adaptive sampling iterations (60, 120 and 170)

The sample locations and the estimated limit surface are shown for different steps of the sampling process, i.e. at iteration 1, 10, 20 and 40 (Figure 13) and 60 120 and 170 (Figure 14) past the training sampling (performed using a 4×4 cartesian grid). For each iteration note how the sample locations are quickly approaching the exact location of the limit surface and the estimated limit surface is converging.

Table 1 compares the number of samples required to evaluate this limit surface by using classical Monte-Carlo and adaptive sampling. Note the much higher number of Monte-Carlo sample that are needed to achieve such low value of convergence.

Table 1 – Number of samples required to evaluate the single region limit surface by using classical Monte-Carlo and adaptive sampling (convergence in value is equal to $5\ 10^{-5}$)

	Number of Samples
Monte-Carlo	$\sim 10^7$
Adaptive	170

7.1.2 Multiple regions

In this second analytical test case we provided a more challenging test case where the limit surface lies not in one but in multiple regions. More specifically, the limit surface in the top right and bottom left corner of a 2-dimensional space. The analytical limit surface is shown in Figure 15. The scope of this test is to show how the sampling process is able to identify limit surfaces that are topologically more complex than the one presented in section 7.1.1.

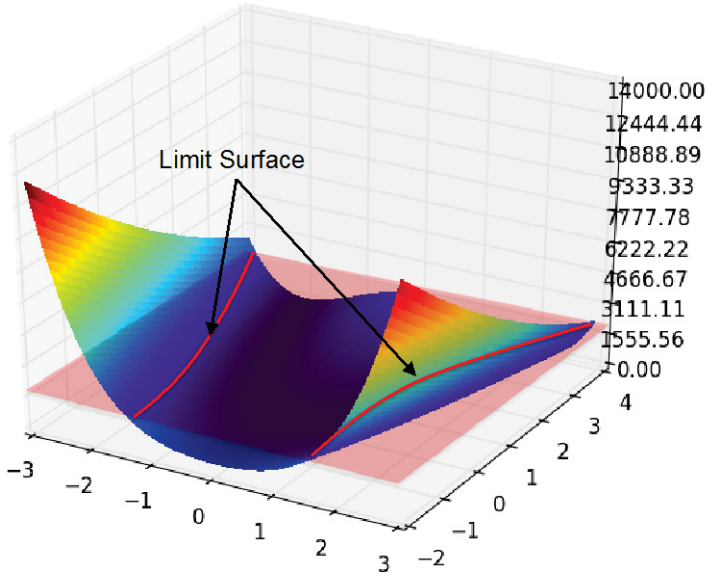

Figure 15 – Analytical shape of the multiple regions limit surface

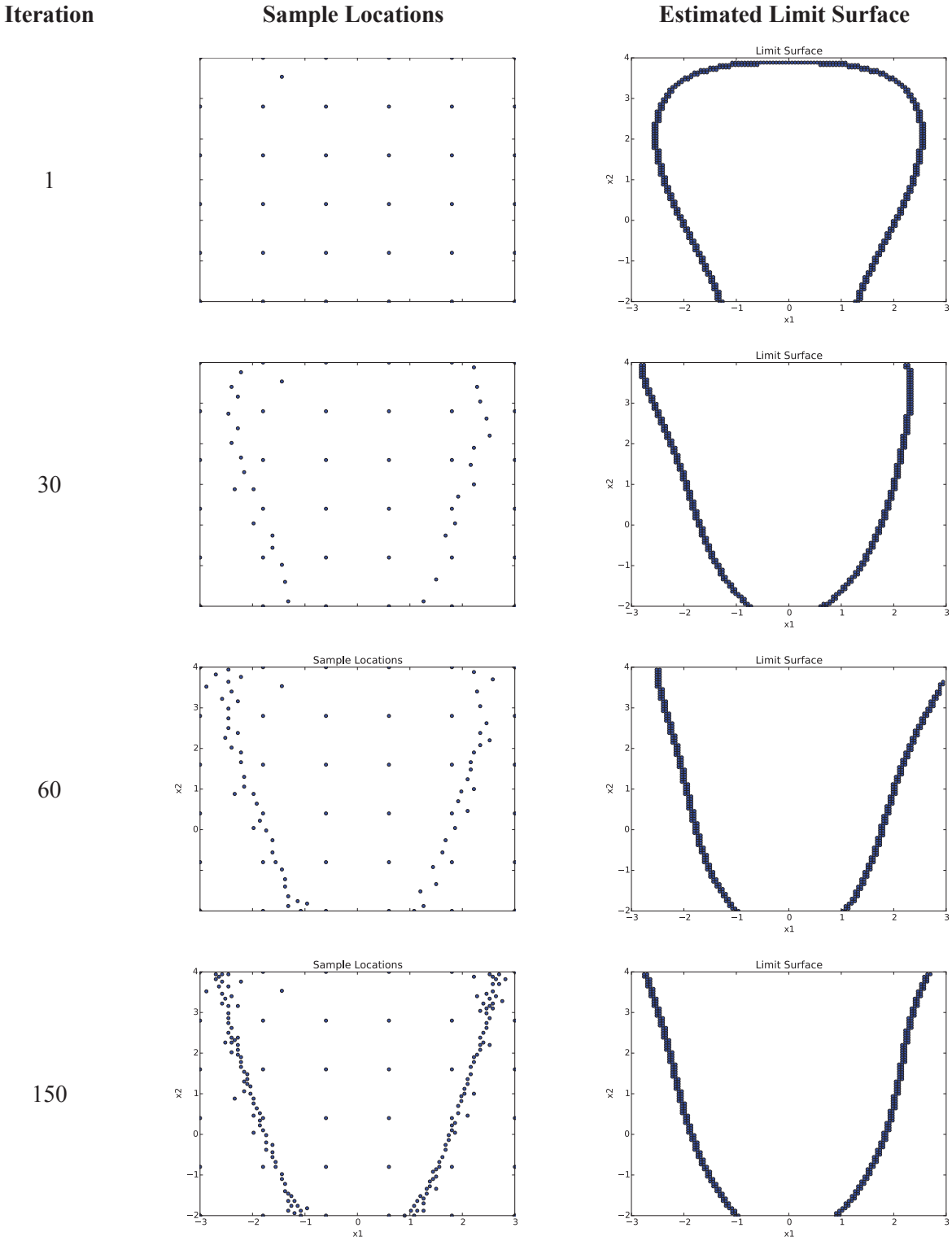

Figure 16 – Multiple regions limit surface: sample locations and the estimated limit surface for different adaptive sampling iterations (1, 30, 60 and 150)

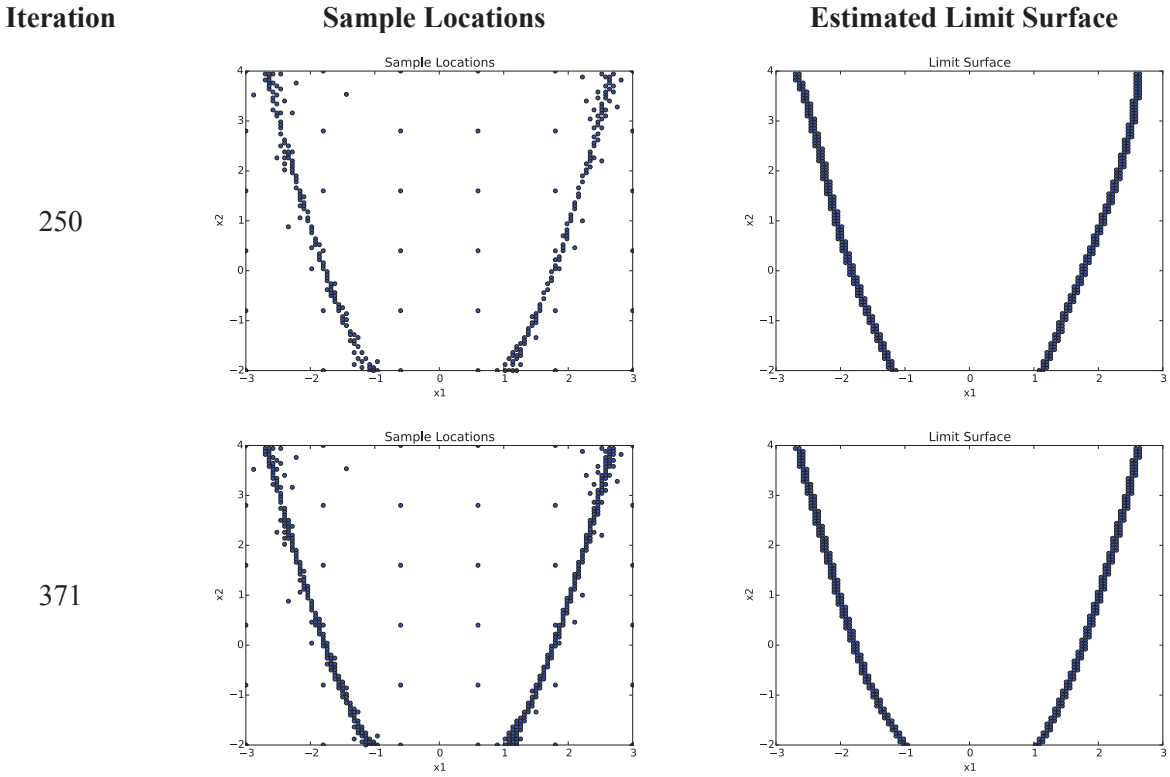

Figure 17 – Multiple regions limit surface: sample locations and the estimated limit surface for different adaptive sampling iterations (250 and 371)

The sample locations and the estimated limit surface are shown for different steps of the sampling process, i.e. at iteration 1, 30, 60 and 1500 (Figure 16) and 250 and 371 (Figure 17) past the training sampling (performed using a 6×6 cartesian grid). For each iteration note how the sample locations are quickly approaching the exact location of the limit surface and the estimated limit surface is converging. Note that for such complex limit surface the required number of samples has increased compared to the case shown in Section 7.1.1: 371 vs. 170.

Table 2 compares the number of samples required to evaluate this limit surface by using classical Monte-Carlo and adaptive sampling.

Table 2 – Number of samples required to evaluate the multiple region limit surface by using classical Monte-Carlo and adaptive sampling (convergence in value is equal to $5\ 10^{-5}$)

	Number of Samples
Monte-Carlo	$\sim 10^7$
Adaptive	371

7.1.3 Convex

The third analytical case aims to test the adaptive sampling algorithms to identify convex limit surfaces (i.e., islands). The analytical limit surface is the result of the intersection of the 3-dimensional surface (see Figure 18):

$$y = x_1^2 + x_2^2 - 0.5 \qquad \text{Eq. 18}$$

along with the plane $y = 0$ (red plane in Figure 18). The resulting limit surface is shown as the red line in Figure 18.

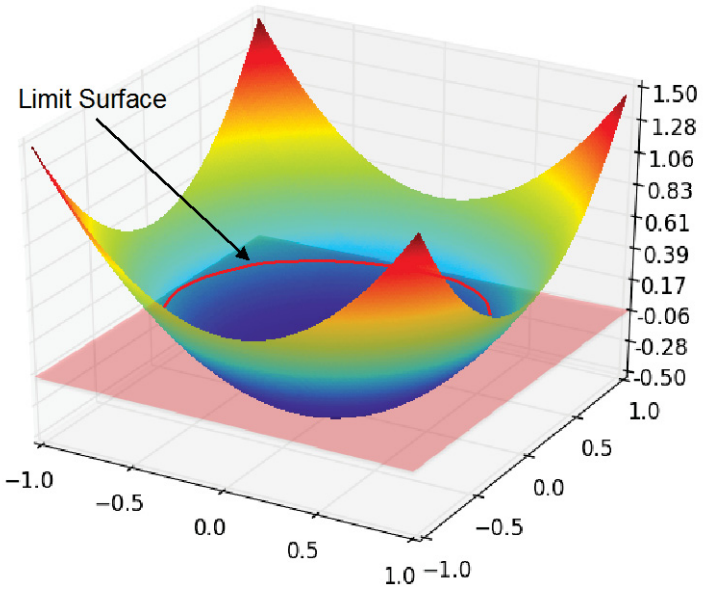

Figure 18 – Analytical shape of the convex limit surface

We performed the adaptive sampling analysis for this test case following an initial 6×6 Cartesian grid sampling for training. The sample locations and the estimated limit surface are shown for different steps of the sampling process, i.e. at iteration 1, 30, 60 and 150 (Figure 19) and 250, 350 and 456 (Figure 20) past the training sampling. For each iteration, note how the sample locations are quickly approaching the exact location of the limit surface and the estimated limit surface is converging.

Note that for such complex limit surface the required number of samples has increased compared to the case shown in Section 7.1.1: 456 vs. 170. Again note such number of samples could drastically decrease for less stringent constraints on convergence criteria.

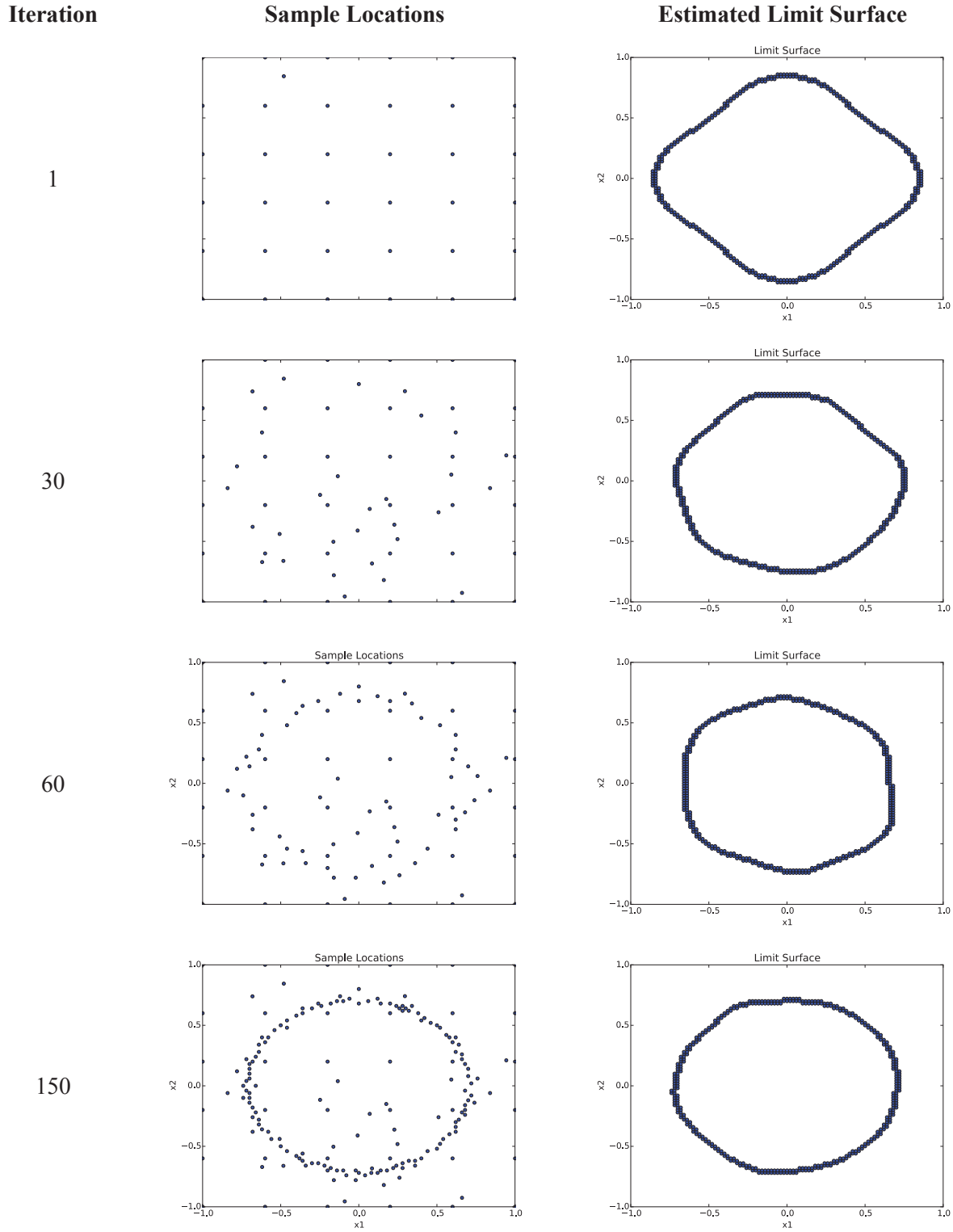

Figure 19 – Convex limit surface: sample locations and the estimated limit surface for different adaptive sampling iterations (1, 30, 60 and 150)

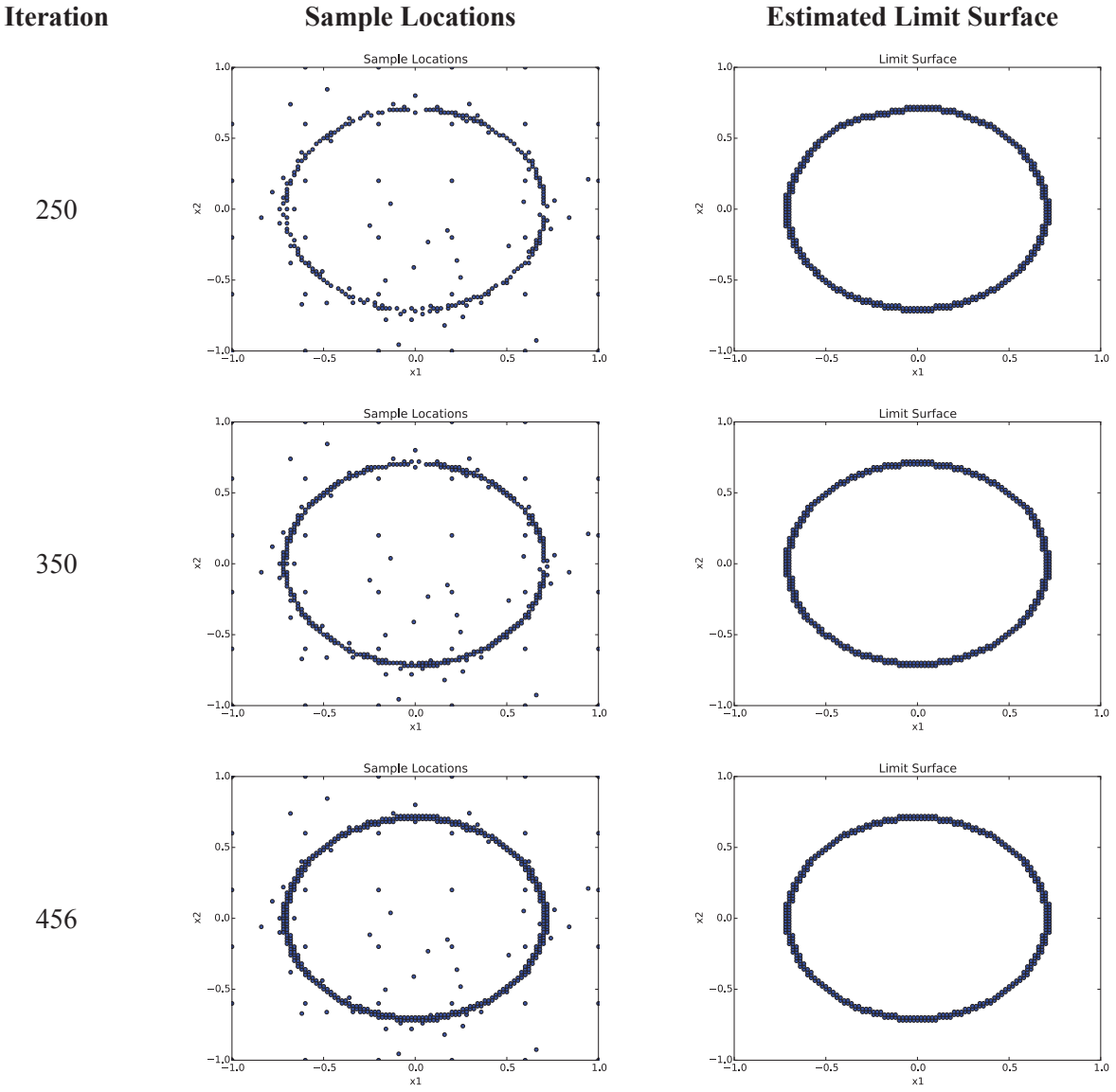

Figure 20 – Convex region limit surface: sample locations and the estimated limit surface for different adaptive sampling iterations (250, 350 and 456)

Table 3 compares the number of samples required to evaluate this limit surface by using classical Monte-Carlo and adaptive sampling.

Table 3 – Number of samples required to evaluate the convex limit surface by using classical Monte-Carlo and adaptive sampling (convergence in value is equal to $5\ 10^{-5}$)

	Number of Samples
Monte-Carlo	$\sim 10^7$
Adaptive	456

7.2 RELAP-7 Test Case

The fourth case presented to test the adaptive sampling scheme uses a more realistic application of adaptive sampling using the RISMC toolkit. Here we employed the RELAP-7 PWR system as model coupled to RAVEN to perform adaptive sampling testing.

The scenario considered is a grid-related loss of off-site power (LOOP). In more detail, the scenario is the following (see Figure 21):

1. An external event (i.e., earthquake) causes the disruption in the power grid and causes a LOOP initiating event; the reactor successfully scrams and, thus, the power generated in the core follows the characteristic exponential decay curve

2. The DGs successfully start and emergency cooling to the core is provided by the Emergency Core Cooling System (ECCS)

3. At a certain time the DGs fail and, thus, conditions of SBO are reached; ECCS systems is subsequently off-line. Without the ability to cool the reactor core, its temperature starts to rise

4. In order to recover AC electric power, a plant recovery team is assembled in order to recover one of the two DGs

5. If AC power is recovered prior reaching code damage condition (CD), the auxiliary cooling system (i.e., ECCS system) is able to cool the reactor core and, thus, core temperature decreases

In this case, we limit the analysis to two stochastic variables:

1. Time of loss of diesel generators (DGs) after LOOP

2. Recovery time of DGs

The RELAP-7 PWR model has been set up based on the parameters specified in the OECD main steam line break (MSLB) benchmark problem [29]. The reference design for the OECD MSLB benchmark problem is derived from the reactor geometry and operational data of the TMI-1 Nuclear Power Plant (NPP), which is a 2772 MW two loop pressurized water reactor (see the system scheme shown in Figure 21).

An example of PWR SBO scenario generated using RELAP-7 is shown in Figure 22

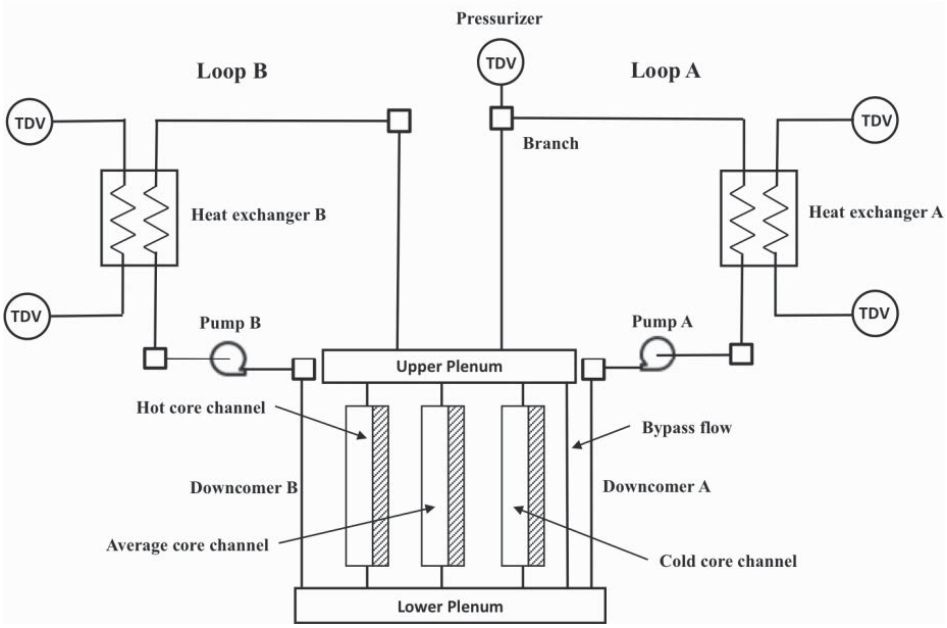

Figure 21 – Scheme of the TMI PWR benchmark

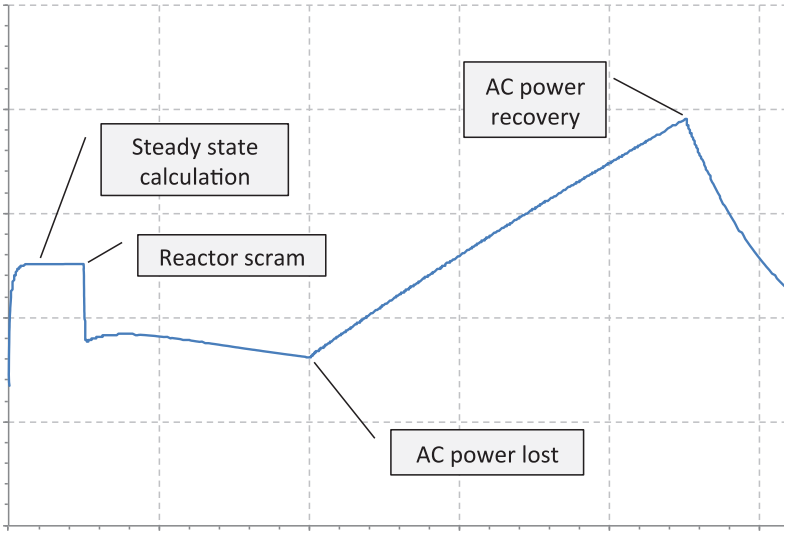

Figure 22 – Example of LOOP scenario followed by DGs failure to run using the RELAP-7 code

For the scope of this report we wanted to show one of the capabilities of RAVEN to generate ROM and perform statistical analysis on them. For this case we collected the actual simulated data by RELAP-7 in [30], generated a ROM from such data and performed adaptive sampling on the ROM instead of the RELAP-7 code. In more detail, we performed the following steps:

1. Retrieved the hdf5 data generated by sampling RELAP-7 in [30]
2. Trained a ROM given the data retrieved in Step 1
3. Sampled on a 2-dimensional Cartesian grid the ROM obtained in Step 2
4. Performed adaptive sampling and limit-surface search

7.2.1 PWR SBO Limit surfaces

We performed the adaptive sampling analysis for this test case following an initial 6×6 Cartesian grid sampling for training. The sample locations and the estimated limit surface are shown for different steps of the sampling process, i.e. at iteration 1, 10, 30 and 60 (Figure 23) and 100, 150 and 185 (Figure 24) past the training sampling. For each iteration note how the sample locations are quickly approaching the exact location of the limit surface and the estimated limit surface is converging.

Table 4 compares the number of samples required to evaluate this limit surface by using classical Monte-Carlo and adaptive sampling.

Table 4 – Number of samples required to evaluate the RELAP-7 PWR SBO limit surface by using classical Monte-Carlo and adaptive sampling (convergence in value is equal to $5\ 10^{-5}$)

	Number of Samples
Monte-Carlo	$\sim 10^7$
Adaptive	185

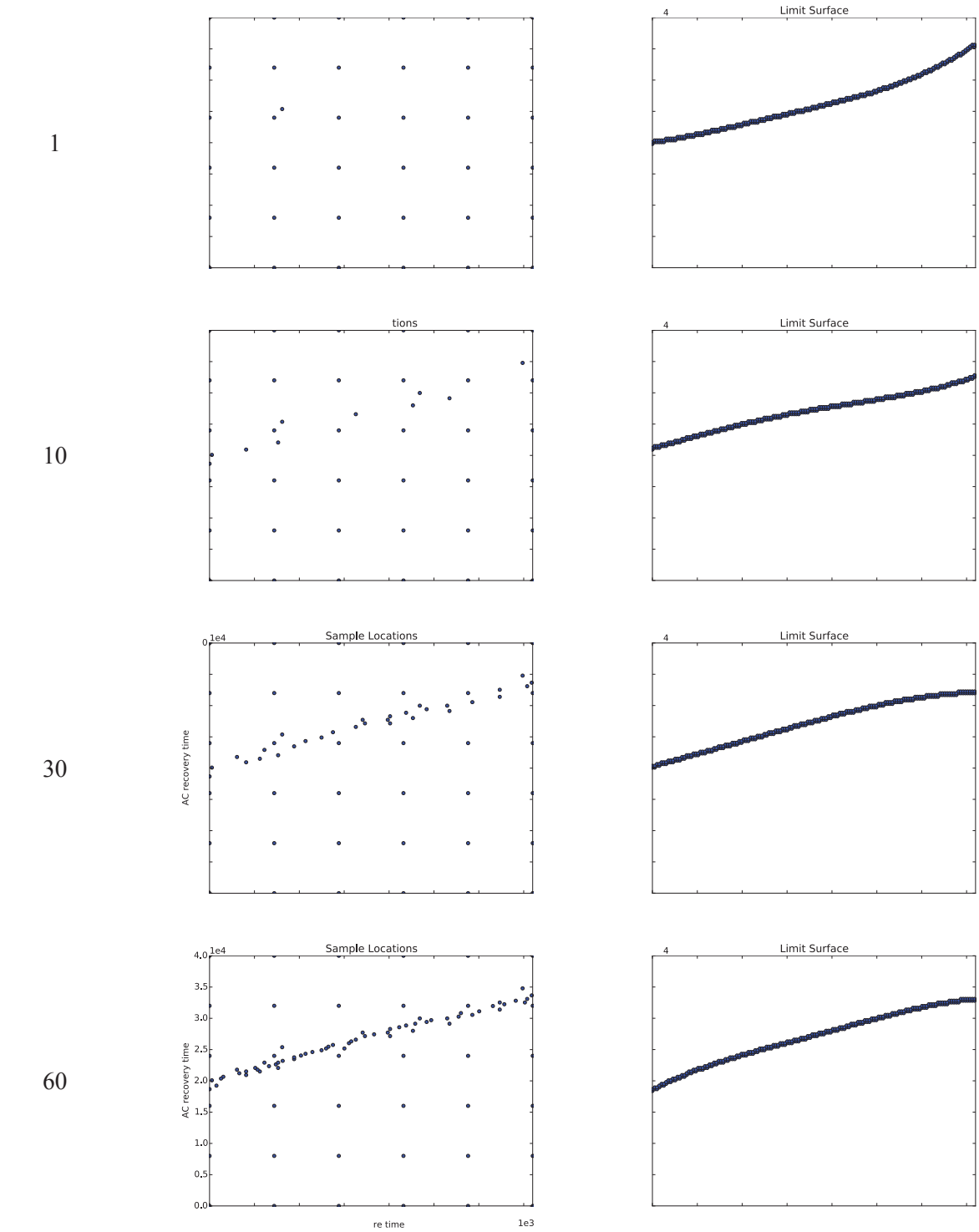

Figure 23 – RELAP-7 limit surface: sample locations and the estimated limit surface for different adaptive sampling iterations (1, 10, 30 and 60)

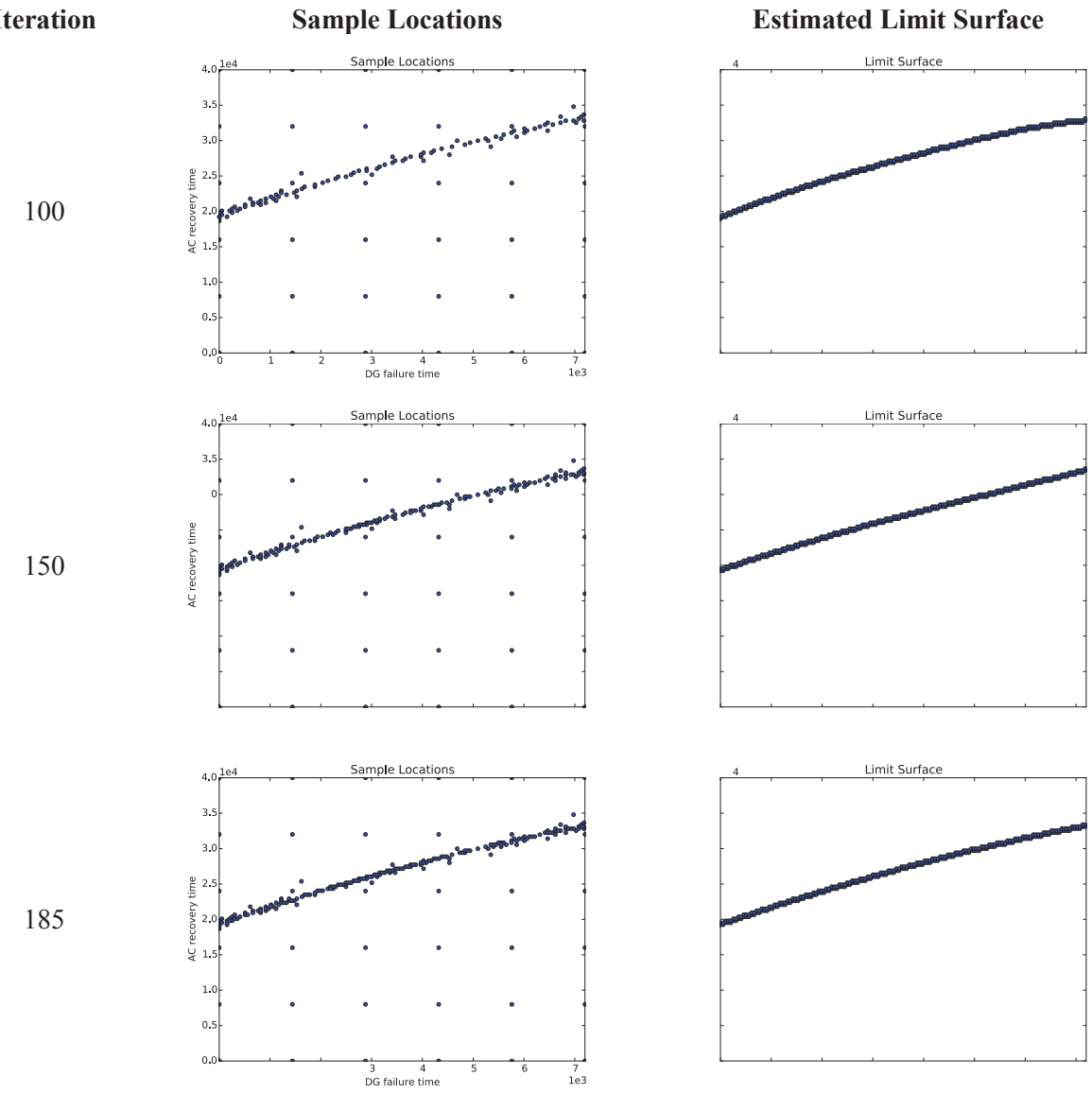

Figure 24 – RELAP-7 limit surface: sample locations and the estimated limit surface for different adaptive sampling iterations (100, 150 and 185)

7.2.2 Limit surface for 100%-120% core power levels

We repeated the analysis of Section 7.1.1 for the case where the reactor power is set to 120% (i.e., a 20% power uprate). The scope of evaluating this new limit surface is to determine the reduction of the time to recovery for DGs. A 20% reactor power increase implies that clad temperature is increasing at a higher rate and thus the clad is reaching its melting temperature (2200 F) much faster.

We performed Steps 1 though 4 for the new data set and evaluated the new limit surface for the 120% test case and the results are shown in Figure 25.

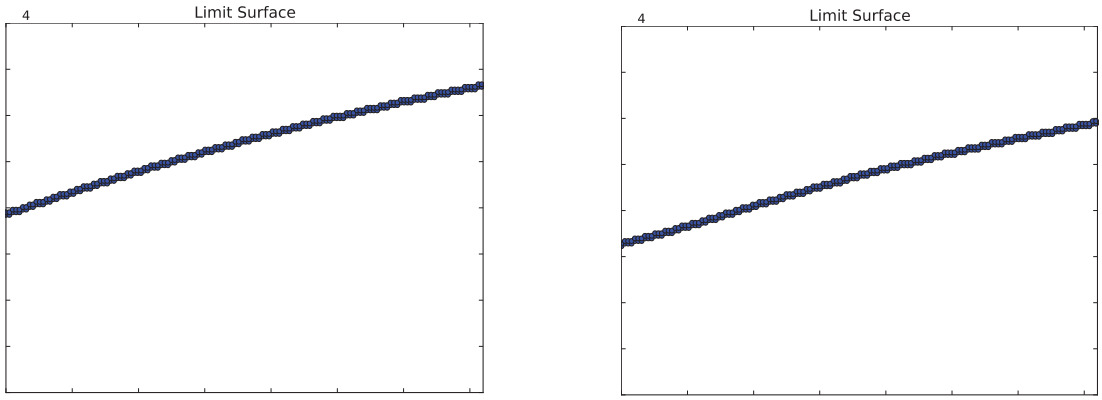

Figure 25 – Limit surface obtained for two different levels of core power: 100% (left) and 120% (right)

7.2.3 PWR SBO probability calculation

In order to show some practical aspects regarding adaptive sampling algorithms and limit surface generation we will show an example regarding the re-calculation of CD probability when the pdf associated to one or more stochastic parameters is changed.

This can be done without re-running the adaptive sampling analysis but by recalculating the probability associated to the sampled points. The obtained limit surface has, per se, a pure deterministic information, i.e. it is independent from the distribution associated to each stochastic parameter.

The probabilistic information associated to the limit surface is generated when the integral $P_{CD} = \int_{\substack{failure \\ region}} pdf(\varpi)\, d\varpi$ of Eq. 8 is calculated. The failure region is confined by the limit surface itself while $pdf(\varpi)\, d\varpi$ is dependent on the distribution associated to each stochastic parameter.

An example is shown in Figure 26 where we employed the analysis shown in Section 7.2.1 for the 100% core power case. Initially, the distributions associated to the two stochastic parameters were the following:

- DG failure time (h): exponential (lambda = 1.09 10-3)
- AC recovery time (h): Weibull (alpha=0.745 and beta=6.14)

In this new test case we assumed that the AC recovery time pdf is changed to take into account the possibility of early recovery of AC power to FLEX-like systems. Thus, the pdf of the two stochastic parameters have been changed with the following ones:

- DG failure time (h): exponential (lambda = 1.09 10-3)
- AC recovery time (h): lognormal distribution (mean=15.0 and sigma=15.0)

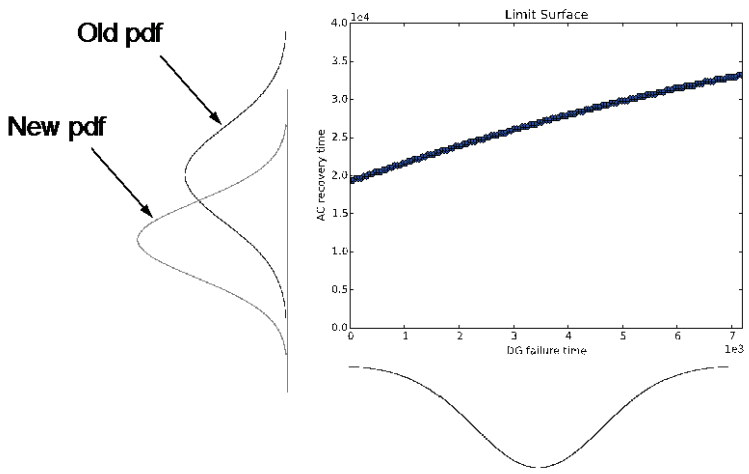

Figure 26 – PWR SBO probability calculation for different pdfs of AC recovery time

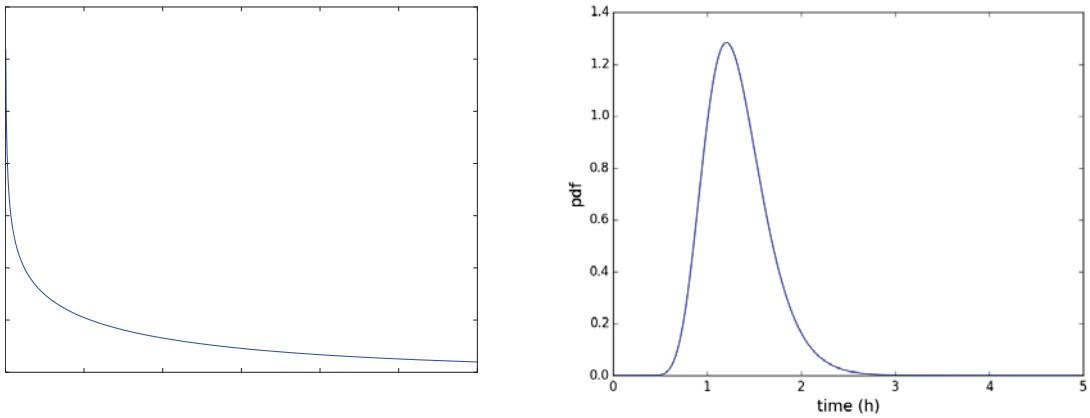

Figure 27 – Original (left) and updated (right) pdfs for AC recovery time

The recalculation was performed by using the limit surface previously obtained in Section 7.2.1. Note that if RELAP-7 was employed the computational costs would have been much higher (orders of magnitude) which proves the validity of the usage of ROMs in place of the system simulator codes. The difference in condition core damage (CCD) probability is shown in Table 5.

Table 5 – Conditional Core Damage Probability (CCDP) for two difference values of pdfs associated to AC recovery time: without (Original) and with (Updated) FLEX-like system

	CCD Probability
Original	$1.2\ 10^{-4}$
Updated	$3.35\ 10^{-11}$

8. Conclusions

In this report we have given an overview of adaptive sampling techniques that can be used to perform PRA analyses using the RISMC toolkit. Classical simulation based approaches rely on either stochastic (e.g., Monte-Carlo or LHS) or deterministic (e.g., Dynamic Event Tree) sampling. As part of the RISMC Pathway, the type of results that can be obtained via simulation goes beyond the evaluation of probability of occurrence of certain events such core damage and containment breach. The RISMC approach aims to determine observable outcomes in order to understand potential vulnerabilities and the limitations of the system under consideration. In order to do so, there is a need to deeply explore the space of possible events. For complex systems such as nuclear power plants, such exploration may require a large number of computationally expensive simulation runs which can be infeasible unless very large high-performance computing resources are used.

Adaptive sampling techniques aim to reduce the computational costs of this kind of analysis by carefully selecting what are the most meaningful simulation runs to be performed. We have shown how such reduction can be achieved for both analytical and more complicated cases. In addition we have shown the kind of information that can be obtained by employing system simulator codes (e.g., RELAP-7) and stochastic analysis tools (e.g., RAVEN) that is unavailable if classical PRA tools (event-tree and fault-tree based) are used.

Classical PRA tools give a limited representation of the system under consideration, for example the timing and sequencing of events is only loosely considered. In [31] we have performed a comparison on classical and RISMC PRA analyses for a BWR SBO test case and we have shown not only the greater amount of information that can be obtained using the RISMC approach but also major differences regarding probability of occurrences of certain event sequences.

Potential computational cost of the RISMC approach is even more significant when multiple codes are needed to simulate a single accident sequence. For example, Figure 28 shows that there are multiple "analysis layers" that may need to be considered when simulating an accident sequence. As an example, a tsunami induced flooding and a consequent SBO condition requires a detailed simulation regarding:

- Seismic energy propagation through the soil and structures (regional level)
- Consequential flooding both on the facility site and (potentially) inside buildings (site topography level)
- Response of the plant structures and components to ground acceleration and flooding (plant level)
- Accident evolution of each unit of the plant (unit level)

Thus, as part of the RISMC approach it is needed to re-think the concept of accident progression through multiple analysis levels. Further note that these analysis levels share information, resources and constraints, i.e., they are tightly coupled. The 2011 Fukushima accident has shown the importance of the interactions among these levels. Compared to classical PRA tools, the external event modeling is not simply an initial condition to each accident sequence but it is actually a time-dependent boundary condition, i.e., the external event evolution progress in parallel to the plant accident evolution.

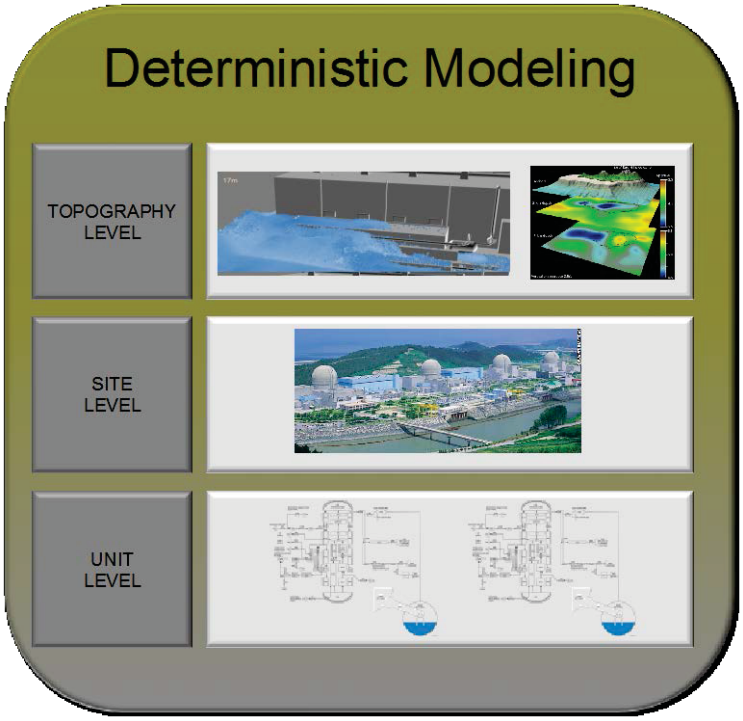

Figure 28 – Multi-layer PRA

It is important to realize not only the computational cost of each simulation run, but also the amount of data generated during the analysis since, for each event sequence, multiple codes are being employed. This rich data summarizes both the inter-levels and intra-levels interactions pictured in Figure 28 which are significant to understand from a safety point of view. Thus, as part of future research, it is needed to develop data analysis algorithms able to extract useful information from large amount of simulation-produced data.

References

[1] C. Smith, C. Rabiti, and R. Martineau, "Risk Informed Safety Margins Characterization (RISMC) Pathway Technical Program Plan", Idaho National Laboratory technical report: INL/EXT-11-22977 (2011).

[2] D. Mandelli, C. Smith, T. Riley, J. Nielsen, J. Schroeder, C. Rabiti, A. Alfonsi, J. Nielsen, R. Kinoshita, D. Maljovec, B. Wang, and V. Pascucci, "Overview of new tools to perform safety analysis: BWR station black out test case," in Proceedings for PSAM 12 Conference, Honolulu (2014).

[3] D. Mandelli, C. Smith, C. Rabiti, A. Alfonsi, R. Youngblood, V. Pascucci, B.Wang, D. Maljovec, P.-T. Bremer, T. Aldemir, A. Yilmaz, and D. Zamalieva, "Dynamic PRA: an overview of new algorithms to generate, analyze and visualize data," *in Proceeding of American Nuclear Society (ANS)*, Washington DC (2013).

[4] H. S. Abdel-Khalik, Y. Bang, J. M. Hite, C. B. Kennedy, C. Wang, "Reduced Order Modeling For Nonlinear Multi-Component Models," *International Journal on Uncertainty Quantification*, **2** - 4, pp. 341-361 (2012).

[5] A. David, R. Berry, D. Gaston, R. Martineau, J. Peterson, H. Zhang, H. Zhao, L. Zou, "RELAP-7 Level 2 Milestone Report: Demonstration of a Steady State Single Phase PWR Simulation with RELAP-7," Idaho National Laboratory technical report: INL/EXT-12-25924 (2012).

[6] S. Prescott, C. Smith, R. Sampath, "Incorporating Dynamic 3D Simulation into PRA", in *ANS PSA 2015 International Topical Meeting on Probabilistic Safety Assessment and Analysis Columbia, SC,* on CD-ROM, American Nuclear Society, LaGrange Park, IL, 2015

[7] R. L. Boring, R. Benish Shirley, J. C. Joe, D, Mandelli, and C. Smith, "Simulation and Non-Simulation Based Human Reliability Analysis Approaches", Idaho National Laboratory technical report: INL/EXT-14-33903 (2014).

[8] 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 technical report: INL/EXT-13-29510 (2013).

[9] D. Gaston, 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).

[10] C. Rabiti, D. Mandelli, A. Alfonsi, J. Cogliati, R. Kinoshita, D. Gaston, R. Martineau, and C. Smith, "RAVEN: a GUI and an artifcial intelligence engine in a dynamic PRA framework," in Proceeding of American Nuclear Society (ANS), Atlanta (GA), **108**, pp. 533-536 (2013).

[11] B. Spencer, Y. Zhang, P. Chakraborty, S.B. Biner, M. Backman, B. Wirth, S. Novascone, J. Hales, "Grizzly Year-End Progress Report", Idaho National Laboratory technical report: INL/EXT-13-30316 (2013).

[12] E. Zio, M. Marseguerra, J. Devooght, and P. Labeau, "A concept paper on dynamic reliability via Monte Carlo simulation," in *Mathematics and Computers in Simulation*, pp. 47–371 (1998).

[13] J. C. Helton and F. J. Davis, "Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems," *Reliability Engineering & System Safety*, **81** – 1 (2003).

[14] E. Zio, *The Monte Carlo Simulation Method for System Reliability and Risk Analysis*, Springer Series in Reliability Engineering (2013).

[15] D. Maljovec, B. Wang, V. Pascucci, P.-T. Bremer, and D. Mandelli, "Adaptive sampling algorithms for probabilistic risk assessment of nuclear simulations," in *ANS PSA 2013 International Topical Meeting on Probabilistic Safety Assessment and Analysis Columbia, SC*, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2013).

[16] J. J. Vandenkieboom, R. W. Youngbloob, J. C. Lee and W. Kerr, "Reliability quantification of advanced reactor passive safety systems", in *Proceeding of American Nuclear Society (ANS)* **76**, pp. 296-298 (1997).

[17] D. Mandelli, S. Prescott, C. Smith, A. Alfonsi, C. Rabiti, J. Cogliati, R. Kinoshita, "Modeling of a Flooding Induced Station Blackout for a Pressurized Water Reactor Using the RISMC Toolkit," *in ANS PSA 2015 International Topical Meeting on Probabilistic Safety Assessment and Analysis Columbia, SC*, on CD-ROM, American Nuclear Society, LaGrange Park, IL, 2015

[18] C. E. Rasmussen, "Gaussian Processes in Machine Learning", Advanced Lectures on Machine Learning, Lecture Notes in Computer Science 3176. pp. 63–71 (2004).

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

[20] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression", *The American Statistician,* **46** – 3, pp. 175–185 (1992).

[21] R. Trudeau, *Introduction to Graph Theory*, Dover Pubblications New York (1993).

[22] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Minining Knowledge Discovery* **2** – 2, pp. 121–167 (1998).

[23] D. Mandelli and C. Smith, "Adaptive sampling using support vector machines," in *Proceeding of American Nuclear Society (ANS)*, San Diego (CA), **107**, pp. 736-738 (2012).

[24] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," *Proceedings of the 1968 ACM National Conference*, pp. 517–524 (1968).

[25] A. Amendola and G. Reina, "Dylam-1, a software package for event sequence and consequence spectrum methodology," in EUR-924, CEC-JRC. ISPRA: Commission of the European Communities (1984).

[26] 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.

[27] C. W. Gardiner, *Handbook of stochastic methods: for physics, chemistry and the natural sciences*, Springer series in synergetics, 13, Springer (2002).

[28] K. Inaba, *Boost C++ Library Programming*, Shuwa System, ISBN: 4-7980-0786-2 (2004).

[29] "Pressurized Water Reactor Main Steam Line Break (MSLB) Benchmark", Volume I: Final Specifications, NEA/NSC/DOC(99)8.

[30] C. Smith, D. Mandelli, S. Prescott, A. Alfonsi, C. Rabiti, J. Cogliati, and R. Kinoshita, "Analysis of pressurized water reactor station blackout caused by external fooding using the RISMC toolkit," Idaho National Laboratory technical report: INL/EXT-14-32906 (2014).

[31] D. Mandelli, C. Smith, Z. Ma, T. Riley, J. Nielsen, A. Alfonsi, C. Rabiti, and J. Cogliati, "Risk-informed safety margin characterization methods development work," Idaho National Laboratory technical report: INL/EXT-14-33191 (2014).