MANUAL

INL/EXT-16-38178
Revision 3
Printed May 3, 2019

RAVEN Theory Manual

Andrea Alfonsi, Cristian Rabiti, Diego Mandelli, Joshua Cogliati, Congjian Wang,
Paul W. Talbot, Daniel P. Maljovec, Curtis Smith

Prepared by
Idaho National Laboratory
Idaho Falls, Idaho 83415

The Idaho National Laboratory is a multiprogram laboratory operated by
Battelle Energy Alliance for the United States Department of Energy
under DOE Idaho Operations Office. Contract DE-AC07-05ID14517.

Issued by the Idaho National Laboratory, operated for the United States Department of Energy by Battelle Energy Alliance.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

INL/EXT-16-38178
Revision 3
Printed May 3, 2019

RAVEN Theory Manual

Andrea Alfonsi
Cristian Rabiti
Diego Mandelli
Joshua Cogliati
Congjian Wang
Paul W. Talbot
Daniel P. Maljovec
Curtis Smith

Contents

1 Introduction

RAVEN [1] [2] [3] [4] is a software framework that allows the user to perform parametric and stochastic analysis based on the response of complex system codes. The initial development was designed to provide dynamic probabilistic risk analysis capabilities (DPRA) to the thermal-hydraulic code RELAP-7 [5], currently under development at Idaho National Laboratory (INL). Now, RAVEN is not only a framework to perform DPRA but it is a multi-purpose stochastic and uncertainty quantification platform, capable of communicating with any system code.

This report serves as a theoretical manual for selected algorithms implemented within the RAVEN framework. It is intended to provide some theorectical treatments of the selected algorithms in the areas of sensitivity and uncertainty analysis, reduced order modeling, statistical analysus, data mining and DPRA that can help the user to understand the theory behind the key concepts.

2 RAVEN Mathematical Background

2.1 System and Control Model

The first step is the derivation of the mathematical model representing, with a high level of abstraction, the plant and control system model. Let $\overline{\theta}(t)$ be a vector describing the system status in the phase space, characterized by the following governing equation:

$$\frac{\partial \overline{\theta}}{\partial t} = \overline{H}\left(\overline{\theta}(t), t\right) \tag{1}$$

In Equation above, the assumption of time differentiability of the trajectory equation $\overline{H}\left(\overline{\theta}(t), t\right)$ in the phase space has been taken. This assumption is not fully correct and generally required and it is used here, without missing of generality, for compactness of the notation.
It can now be performed an arbitrary decomposition of the phase space:

$$\overline{\theta} = \begin{pmatrix} \overline{x} \\ \overline{v} \end{pmatrix} \tag{2}$$

The decomposition is made in such a way that $\overline{x}$ represent the unknowns solved by a system code (such as RELAP5-3D [6], RELAP7 [5], etc.) while $\overline{v}$ are the variables directly controlled by the control system (e.g., automatic mitigation systems, operator actions, etc.).
The governing equation can be now cast in the following system of equations:

$$\begin{cases} \frac{\partial \overline{x}}{\partial t} = \overline{F}(\overline{x}, \overline{v}, t) \\ \frac{\partial \overline{v}}{\partial t} = \overline{V}(\overline{x}, \overline{v}, t) \end{cases} \tag{3}$$

Consequentially to this splitting, $\overline{x}$ contains the state variables of the phase space that are continuous while $\overline{v}$ contains discrete state variables that are usually handled by the control system (consequentially, named **control variables**). It can be noticed that the function $\overline{V}(\overline{x}, \overline{v}, t)$, representing the control system, does not depend on the knowledge of the complete status of the system but on a restricted subset that can be named **monitored variables** $\overline{C}$:

$$\begin{cases} \frac{\partial \overline{x}}{\partial t} = \overline{F}(\overline{x}, \overline{v}, t) \\ \overline{C} = \overline{G}(\overline{x}, t) \\ \frac{\partial \overline{v}}{\partial t} = \overline{V}(\overline{x}, \overline{v}, t) \end{cases} \tag{4}$$

where $\overline{C}$ is a vector of smaller dimensionality than $\overline{x}$ and, therefore, more convenient to handle.
As it can be noticed, the standard nomenclature of *signals* (monitored variables) and *status* (control variables) is not adopted. Two principal reasons justify this decision:

- The definition of signals is tight to the definition of the control logic for each component and, therefore, relative rather than absolute in the overall system analysis. For example, it is possible the the *signals* for a component represent *status* of another one, determining an in-unique definition.

- The standard nomenclature becomes meaningless when this derivation is applied to Uncertainty Quantification (UQ).

2.1.1 Splitting Approach for the Simulation of the Control System

Equation 4 represents a fully coupled system of Partial Differential Equations (PDEs). To solve this system, an *operator splitting* approach is employed. This method is preferable in this context for several reasons, among which the following:

- In reality, the control system (automatic mitigation systems, operator actions, etc.) is always characterized by an intrinsic delay
- The reaction of the control system might make the system "move" among different discrete states; therefore, numerical errors will be always of first order unless the discontinuity is explicitly treated.

Employing the *operator splitting* approach, Equation 4 can be cast as follows:

$$\begin{cases} \frac{\partial \overline{x}}{\partial t} = \overline{F}\left(\overline{x}, \overline{v}_{t_{i-1}}, t\right) \\ \quad \overline{C} = \overline{G}(\overline{x}, t) \\ \frac{\partial \overline{v}}{\partial t} = \overline{V}\left(\overline{x}, \overline{v}_{t_{i-1}}, t\right) \end{cases} \quad t_{i-1} \leq t \leq t_i = t_{i-1} + \Delta t_i \tag{5}$$

Hence, the system of equations in solved decomposing it into simpler sub-problems that are treated individually using specialized numerical algorithms.

2.1.2 Definition of the Monitored Variable Space

The contraction of the information from the $\overline{x}$ space to the $\overline{C}$ space is a crucial step. Since $\overline{C}$ represents an arbitrary middle step, it is needed to define a set of rules that make this choice unique. $\overline{C}$ is chosen such that:

- The solution of $\frac{\partial \overline{v}}{\partial t}\Big| = \overline{V}\left(\overline{x}, \overline{v}_{t_{i-1}}, t\right)$ can be carried along without any knowledge of the solution algorithm of $\frac{\partial \overline{x}}{\partial t} =\Big| \overline{F}\left(\overline{x}, \overline{v}_{t_{i-1}}, t\right)$. This requirement determines the minimum information contraction from $\overline{x}$ to $\overline{C}$.
- All actions represented by $\overline{C} = \overline{G}(\overline{x}, t)$ require knowledge of the solution algorithm of $\frac{\partial \overline{x}}{\partial t} =\Big| \overline{F}\left(\overline{x}, \overline{v}_{t_{i-1}}, t\right)$. This requirement determines the maximum information contraction from $\overline{x}$ to $\overline{C}$.

The intersection of the two sub-spaces defined above create a minimal unique set.

2.1.3 Definition of the Auxiliary Variable Space

In the previous sections, it has been determined that the needed information to model the dynamic system is contained in the solution vectors $\overline{x}$ and $\overline{v}$. Even if $\overline{x}$ and $\overline{v}$ are sufficient to assess the system status at every point in time, it can result in an unpractical way to model the eventual control system. Let's suppose to model a component of a particular system that presents different behavior depending on other systems or operation behaviors. In order to define the status of this component in every point in time, the whole history of the system needs to be tracked. In order to remove these inefficiency, a set of auxiliary variables $\overline{a}$ can be introduced. These variables are the ones that in the analysis of stochastic dynamics are artificially added into the phase space to a non-Markovian system to obtain back a Markovian behavior. In this way only the previous time-step information is needed to determine the status of the system.
Adding this additional system of variables, Equation 5 can be casted as follows:

$$\begin{cases} \frac{\partial \overline{x}}{\partial t} = \overline{F}\left(\overline{x}, \overline{v}_{t_{i-1}}, t\right) \\ \quad \overline{C} = \overline{G}(\overline{x}, t) \\ \frac{\partial \overline{a}}{\partial t} = \overline{A}\left(\overline{x}, \overline{C}, \overline{a}, \overline{v}_{t_{i-1}}, t\right) \\ \frac{\partial \overline{v}}{\partial t} = \overline{V}\left(\overline{C}, \overline{a}, \overline{v}_{t_{i-1}}, t\right) \end{cases} \quad t_{i-1} \leq t \leq t_i = t_{i-1} + \Delta t_i \tag{6}$$

2.2 Dynamic Systems Stochastic Modeling

2.2.1 General system of equations and variable classification

In Section 2.1, the derivation of the governing equations for a controllable system have been reported. In this section, the mathematical framework of the modeling of dynamic stochastic systems, under uncertainties, is derived.
Dynamic stochastic systems are the ones whose dynamic is characterized by intrinsic randomness. Random behaviors, although present in nature, are often artificially introduced into physical models to account for the incapability of fully modeling part of the nature of the system behavior and/or of the phenomena bounding the physical problem.
The distinction between variables that are artificially considered aleatory and the ones intrinsically aleatory corresponds with the classical definition of epistemic and aleatory uncertainties. From a system simulation point of view it is more relevant how these variables, the sources of aleatory behavior, change in time. Possible examples of random elements are:

- random variability of parameters (e.g., uncertainty in physical parameters)
- presence of noise (background noise due to intrinsically stochastic behaviors or lack of detail in the simulation)

- Uncertainty in the initial and boundary conditions
- Random failure of components
- aging effects.

Before introducing the mathematical models for uncertainty, it can beneficial to recall Equation 1, adding the initial conditions:

$$\begin{cases} \frac{\partial \overline{\theta}(t)}{\partial t} = \overline{H}\left(\overline{\theta}\left(t\right), t\right) \\ \overline{\theta}\left(t_0\right) = \overline{\theta}_0 \end{cases} \tag{7}$$

At this point, each source of uncertainty or stochastic behavior is considered and progressively added in Equation 7. For the scope of this derivation, it is convenient to split the phase space into *continuous* (e.g.,temperature, pressure, hentalpy, etc.) and discrete (e.g.,status of components, such as operational and failure states) variables as follows:

- $\overline{\theta}^c \in \Phi \subseteq \mathbb{R}^C$, the set of continuous variables
- $\overline{\theta}^d \in \Psi \subseteq \mathbb{N}^D$, the set of discrete variables
- $\overline{\theta}(t) = \overline{\theta}^c \oplus \overline{\theta}^d$.

Consequentially, Equation 7 assumes the following form:

$$\begin{cases} \frac{\partial \overline{\theta}^c(t)}{\partial t} = f\left(\overline{\theta}^c, \overline{\theta}^d, t\right) \\ \frac{\partial \overline{\theta}^d(t)}{\partial t} = g\left(\overline{\theta}^c, \overline{\theta}^d, t\right) \\ \overline{\theta}^c\left(t_0\right) = \overline{\theta}_0^c \\ \overline{\theta}^d\left(t_0\right) = \overline{\theta}_0^d \end{cases} \tag{8}$$

Note that the time derivative operator has been also used for the time discontinuous variables, even if this is allowed only introducing complex extension of the time derivative operator. In this context, the $\frac{\partial}{\partial t}$ on the discontinuous space is employed for simplifying the notation only.

2.2.2 Probabilistic Nature of the Parameters Characterizing the Equation

As shown in Equation 9, The first stochastic behaviors to be introduced are the uncertainties associated with the:

- initial conditions (i.e. $\overline{\theta}^c$ and $\overline{\theta}^d$ at time t_0), and
- parameters characteristic of $f\left(\overline{\theta}^c, \overline{\theta}^d, t\right)$ and $g\left(\overline{\theta}^c, \overline{\theta}^d, t\right)$.

$$\begin{cases} \frac{\partial \overline{\theta}^c(t)}{\partial t} = f\left(\overline{\theta}^c, \overline{\theta}^d, \overline{\alpha}_{staz}, t\right) \\ \frac{\partial \overline{\theta}^d(t)}{\partial t} = g\left(\overline{\theta}^c, \overline{\theta}^d, \overline{\alpha}_{staz}, t\right) \\ \Pi\left(\overline{\theta}^c, t_0\right) \sim pdf\left(\overline{\theta}^c_0|, \sigma_c^2\right) \\ \Pi\left(\overline{\theta}^d, t_0\right) \sim pdf\left(\overline{\theta}^d_0|, \sigma_d^2\right) \\ \overline{\alpha}_{staz}\left(t\right) = \overline{\alpha}_{staz}\left(t_0\right) \sim pdf\left(\overline{\alpha}^0_{staz}|, \sigma^2_{staz}\right) \end{cases} \tag{9}$$

In Equation 9, $\Pi\left(\overline{\theta}^c, t_0\right)$ indicates the probability distribution of $\overline{\theta}^c$ at the initial time $t = t_0$ while $pdf\left(\mu|, \sigma^2\right)$ represents a generic probability distribution function having mean value μ and sigma σ.The term $\overline{\alpha}_{staz}$ is the vector of parameters affected by uncertainty but not varying over time.
As already mentioned, Equation 9 considers uncertainties whose values do not change during the dynamic evolution of the system. This set of uncertainties accounts for most of the common source of aleatory behaviors. Examples of this kind of uncertainties are:

- *Uncertainty associated with the heat conduction coefficient*: This value is known (but uncertain) and has no physical reason to change during the simulation;
- *Uncertainty on failure temperature for a pipe*: This value is usually characterized by a probability distribution function but once the value has been set (like through random sampling) it will not change during the simulation.

From a modeling perspective, all the probabilistic behaviors connected to $\Pi\left(\overline{\theta}^c, t_0\right)$, $\Pi\left(\overline{\theta}^d, t_0\right)$ and $\overline{\alpha}_{staz}(t)$ can be modeled without changing the dimensionality of the phase space (hence, no alteration of the solution algorithm is required), simply performing sampling of the input space. In addition, the Markovian assumption is still preserved.

2.2.3 Variables Subject to Random Motion

The next aleatory component to be accounted for is the set of parameters that continuously change over time (i.e. $\overline{\alpha}_{brow}$). In other words, these parameters are referred as if they behave like a *Brownian motion*. While what commonly is indicated as *Brownian motion* has not impact at the character the space and time scales (characteristic of a physical system), there are parameters that have (or **appear** to have) such behavior. The *Brownian motion* characteristic of some variables can be completely synthetic, due to the lack of modeling details in the simulation model.
For instance, two examples of these randomly varying variables are:

- *Cumulative damage growth in material*. Experimental data and models representing this phenomenon show large uncertainties. There is also an intrinsic natural stochasticity driving the accumulation of the damage (natural Brownian motion);

- *Heat conductivity in the fuel gap during heating of fuel.* During some transients there are situations where the fuel is in contact with the clad while in others where there is the presence of a gap. While in nature this is a discontinuous transition, it is not usually possible to model in such a way, especially if vibrations of the fuel lead to high frequency oscillations. In this case, it would be helpful to introduce directly into the simulation a random noise characterizing the thermal conductivity when these transitions occur (synthetic Brownian motion).

The system of Equations 9 can be rewritten in the following form:

$$\begin{cases} \frac{\partial \overline{\theta}^c(t)}{\partial t} = f\left(\overline{\theta}^c, \overline{\theta}^d, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) \\ \frac{\partial \overline{\theta}^d(t)}{\partial t} = g\left(\overline{\theta}^c, \overline{\theta}^d, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) \\ \frac{\partial \overline{\alpha}_{brow}}{\partial t} = b\left(\overline{\theta}^c, \overline{\theta}^d, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right)\Gamma(t) \\ \Pi\left(\overline{\theta}^c, t_0\right) \sim pdf\left(\overline{\theta}^c_0|, \sigma_c^2\right) \\ \Pi\left(\overline{\theta}^d, t_0\right) \sim pdf\left(\overline{\theta}^d_0|, \sigma_d^2\right) \\ \overline{\alpha}_{staz}(t) = \overline{\alpha}_{staz}(t_0) \sim pdf\left(\overline{\alpha}^0_{staz}|, \sigma^2_{staz}\right) \\ \overline{\alpha}_{brow}(t_0) \sim \overline{\alpha}^0_{brow}\Gamma(t_0) \end{cases} \tag{10}$$

where $\Gamma(t)$ is 0-mean random noise and $\overline{\alpha}_{brow}$ is the set of parameters subject to *Brownian motion*. Clearly, the equation referring to the time change of the parameters subject to the *Brownian motion* should be interpreted in the **Ito** sense [C. Gardiner, Stochastic Methods, Springer (2009)].

2.2.4 Discontinuously and Stochastically varying variables

The last and probably most difficult step is the introduction of parameters that are neither constant during the simulation nor continuously vary over time. As an example, consider a valve that, provided set of operating conditions, opens or closes. If this set of conditions is reached n times during the simulation, the probability of the valve correctly operating should be sampled n times. It is also foreseeable that the history of failure/success of the valve will impact future probability of failure/success. In this case the time evolution of such parameters (discontinuously stochastic changing parameters $\overline{\alpha}_{DS}$) is governed by the following equation:

$$\frac{\partial \overline{\alpha}_{DS}(t)}{\partial t} = \overline{\delta}\left(\overline{\alpha}_{DS}, \overline{\theta}^c, \overline{\theta}^d, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) \times \overline{V}\left(\overline{\alpha}_{DS}, \overline{\theta}^c, \overline{\theta}^d, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) \times \overline{p}\left(\int_{t_0}^{t} S\left(\overline{\theta}\left(t'\right), t'\right) dt'\right) \tag{11}$$

where:

- The function $\overline{\delta}\left(\overline{\alpha}_{DS}, \overline{\theta}^{c}, \overline{\theta}^{d}, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right)$ is the delta of Dirac of the instant on which the transition need to be evaluated (control logic signaling to the valve to open/close).
- The term $\overline{p}\left(\int_{t_0}^{t} S\left(\overline{\theta}\left(t^{'}\right), t^{'}\right)\right) = \overline{p}\left(\int_{t_0}^{t} \overline{\alpha}_{DS}, \overline{\theta}^{c}, \overline{\theta}^{d}, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, tdt\right)$ represents the transition probability between different states (in case of the valve: open/close). Note that the time integral of the parameter history accounts for the memory of the component via the kernel $S\left(\overline{\theta}\left(t^{'}\right), t^{'}\right)$.
- The term $\overline{V}\left(\overline{\alpha}_{DS}, \overline{\theta}^{c}, \overline{\theta}^{d}, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right)$ is the rate of change of $\overline{\alpha}_{DS}$. For a discrete parameter, it is defined as the value of the instantaneous $\overline{\alpha}_{DS}$ change.

The introduction of the history dependency introduced in the term $\overline{p}$ determines that the system cannot be considered Markovian if "countermeasures" are not taken. In order to make the system return to be Markovian, the phase space needs to be expanded (i.e., increase its dimensionality): the time at which the parameters changed status and their corresponding values $\{(\overline{\alpha}_{DS}, t)_i\} = \{\overline{\alpha}_{DS}, t_i\} = \overline{\overline{\alpha}}_{DS}, \overline{t}\ (for\, i = 1, ..., n)$.
Equation 11 now assumes the form:

$$\frac{\partial \overline{\alpha}_{DS}(t)}{\partial t} = \overline{\delta}\left(\overline{\alpha}_{DS}, \overline{\theta}^{c}, \overline{\theta}^{d}, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) \times \overline{V}\left(\overline{\alpha}_{DS}, \overline{\theta}^{c}, \overline{\theta}^{d}, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) \times \overline{p}\left(\overline{\overline{\alpha}}_{DS}, \overline{t}, \overline{\theta}^{c}, \overline{\theta}^{d}, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) \quad for\, t \geq t_n \tag{12}$$

This formulation introduces a phase space that is continuously growing over time $n \to \infty$. In this respect, it is useful to introduce and discuss possible assumptions:

1. The memory of the past is not affected by the time distance; in this case:

$$\overline{p}\left(\overline{\overline{\alpha}}_{DS}, \overline{t}, \overline{\theta}^{c}, \overline{\theta}^{d}, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) = \overline{p}\left(\overline{\overline{\alpha}}_{DS}, \overline{\theta}^{c}, \overline{\theta}^{d}, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) \tag{13}$$

   The dimensionality of the phase space is still growing during the simulation since more and more sampling is performed, but the time integral is removed from the transition probability. A simple example of this situation is a component activated on demand in which failure is a function of all previous sampling, but not of when the component was sampled or in which sequence the outcome occurred.

2. The number of samples is determined before the simulation itself takes place (e.g.,n times) In this case the different $\overline{\alpha}_{DS_i}$ could be treated explicitly as $\overline{\alpha}_{staz}$ while $\overline{t}$ would still remain a variable to be added to the phase space (if simplification 1 is not valid) but of fixed dimension. In this case $\overline{t}$ still needs to be computed and its expression is:

$$\overline{t}\,(t) = \int_{t_0}^{t} \overline{t}\,\overline{\delta}\left(\overline{\alpha}_{DS}, \overline{\theta}^{c}, \overline{\theta}^{d}, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) dt \tag{14}$$

The transition probability becomes:

$$\overline{p}\left(\int_{t_0}^{t} dt\, S\left(\overline{t}\right), \overline{\alpha}_{DS}, \overline{\theta}^{c}, \overline{\theta}^{d}, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) \tag{15}$$

For example, this is the case of a component that is sampled a fixed number of times for a given simulation while the contribution of the history to the transition probability might decay exponentially over time. This approximation might eliminate the memory from the system by adding n variables to the phase space t_i $(for\ i = 1, ..., n)$ thus restoring the Markovian characteristic.

3. Another possible approximation alternative to the previous one is that the memory of the system (here explicitly represented by $\int_{t_0}^{t} \overline{\alpha}_{DS} dt$) is limited only to a fixed number of steps back in the past. In this case n is always bounded. Therefore, adding $\{\overline{\alpha}_{DS_i}, t_i\}, (for\ i = 1, ..., n)$ would possibly preserve the system Markovian properties of the system. This approximation allows for eliminating the memory from the system by expanding the phase space $2n$ variables. From a software implementation point of view, this is the most complex situation since without any simplification we would have to deal with a system that is never reducible to a Markovian one and therefore forced to use the whole history of the system to forecast its evolution at each time step.

Assumption 1 limits this cost by restraining it to the set of values assumed by the variable but would still lead to very difficult to deal with situation. Assumption 2 would require an expansion of phase space to introduce the time at which the transitions happens but the value that the parameter will assume at each sampling could be treated as initial condition. Assumption 3 would instead require the expansion of the phase space for both the time and the values of the transitioning variables.
Based on the this simplifications, the system of Equations 10, accounting also for $\overline{\alpha}_{DS}$ can be cast into the form:

$$\begin{cases} \frac{\partial \overline{\theta}^{c}(t)}{\partial t} = f\left(\overline{\theta}^{c}, \overline{\theta}^{d}, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) \\ \frac{\partial \overline{\theta}^{d}(t)}{\partial t} = g\left(\overline{\theta}^{c}, \overline{\theta}^{d}, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) \\ \frac{\partial \overline{\alpha}_{brow}}{\partial t} = b\left(\overline{\theta}^{c}, \overline{\theta}^{d}, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) \Gamma\left(t\right) \\ \frac{\partial \overline{\alpha}_{DS}(t)}{\partial t} = \overline{\delta}\left(\overline{\alpha}_{DS}, \overline{\theta}^{c}, \overline{\theta}^{d}, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) \times \overline{V}\left(\overline{\alpha}_{DS}, \overline{\theta}^{c}, \overline{\theta}^{d}, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) \times \\ \times \overline{p}\left(\int_{t_0}^{t} dt\, \overline{\alpha}_{DS}, \overline{\theta}^{c}, \overline{\theta}^{d}, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) \\ \Pi\left(\overline{\theta}^{c}, t_0\right) \sim pdf\left(\overline{\theta}_0^{c}|, \sigma_c^2\right) \\ \Pi\left(\overline{\theta}^{d}, t_0\right) \sim pdf\left(\overline{\theta}_0^{d}|, \sigma_d^2\right) \\ \overline{\alpha}_{staz}\left(t\right) = \overline{\alpha}_{staz}\left(t_0\right) \sim pdf\left(\overline{\alpha}_{staz}^{0}|, \sigma_{staz}^2\right) \\ \overline{\alpha}_{brow}\left(t_0\right) \sim \overline{\alpha}_{brow}^{0} \Gamma\left(t_0\right) \\ \overline{\alpha}_{DS}\left(t_0\right) = \overline{\alpha}_{DS}^{0} \end{cases} \tag{16}$$

Introducing the Simplifications **1** and **3** (the most appropriated in this context), Equation 16 becomes:

$$\begin{cases} \frac{\partial \overline{\theta}^c(t)}{\partial t} = f\left(\overline{\theta}^c, \overline{\theta}^d, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) \\ \frac{\partial \overline{\theta}^d(t)}{\partial t} = g\left(\overline{\theta}^c, \overline{\theta}^d, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) \\ \frac{\partial \overline{\alpha}_{brow}}{\partial t} = b\left(\overline{\theta}^c, \overline{\theta}^d, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right)\Gamma\left(t\right) \\ \frac{\partial \overline{\alpha}_{DS}(t)}{\partial t} = \overline{\delta}\left(\overline{\alpha}_{DS}, \overline{\theta}^c, \overline{\theta}^d, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) \times \overline{V}\left(\overline{\alpha}_{DS}, \overline{\theta}^c, \overline{\theta}^d, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) \times \\ \times \overline{p}\left(\overline{\alpha}_{DS}, \overline{\theta}^c, \overline{\theta}^d, \overline{\alpha}_{staz}, \overline{\alpha}_{brow}, t\right) \\ \Pi\left(\overline{\theta}^c, t_0\right) \sim pdf\left(\overline{\theta}_0^c|, \sigma_c^2\right) \\ \Pi\left(\overline{\theta}^d, t_0\right) \sim pdf\left(\overline{\theta}_0^d|, \sigma_d^2\right) \\ \overline{\alpha}_{staz}\left(t\right) = \overline{\alpha}_{staz}\left(t_0\right) \sim pdf\left(\overline{\alpha}_{staz}^0|, \sigma_{staz}^2\right) \\ \overline{\alpha}_{brow}\left(t_0\right) \sim \overline{\alpha}_{brow}^0 \Gamma\left(t_0\right) \\ \overline{\alpha}_{DS}\left(t_0\right) = \overline{\alpha}_{DS}^0 \end{cases} \tag{17}$$

This dissertation does not cover all the possible phenomena, but it provides a sufficient mathematical framework for extrapolating toward cases that are not explicitly treated.
Given the presence of all these sources of stochastic behaviors, every exploration of the uncertainties (through sampling strategies) only represents a possible trajectory of the system in the phase space. Hence, it is much more informative the assessment of the probability of a particular response, rather than the response itself.
The explanation of these concepts is demanded to next section.

2.3 Formulation of the equation set in a statistical framework

Based on the premises reported in the previous sections and assuming that at least one of the simplifications mentioned in Section 2.2.1 is applicable (i.e. the system can be casted as Markovian), it is needed to investigate the system evolution in terms of its probability density function in the global phase space $\overline{\theta}$ via the Chapman-Kolmogorov equation [7].
The integral form of the Chapman-Kolmogorov is the following:

$$\Pi\left(\overline{\theta}_3, t_3|\overline{\theta}_1, t_1\right) = \int d\overline{\theta}_2 \Pi\left(\overline{\theta}_2, t_2|\overline{\theta}_1, t_1\right) \Pi\left(\overline{\theta}_3, t_3|\overline{\theta}_2, t_2\right) \quad where\ t_1 < t_2 < t_3 \tag{18}$$

while its differential form is:

$$\frac{\partial \Pi\left(\overline{\theta}, t|\overline{\theta}_0, t_0\right)}{\partial t} = \mathcal{L}_{CK}\left(\Pi\left(\overline{\theta}, t|\overline{\theta}_0, t_0\right)\right) \tag{19}$$

The transition from the integral to the differential form is possible under the following assumptions:

$$\lim_{\Delta t \to 0} \frac{1}{\Delta t} \int_{|\overline{\theta}_2 - \overline{\theta}_1| < \varepsilon} \Pi\left(\overline{\theta}_2, t + \Delta t|\overline{\theta}_1, t\right) d\overline{\theta}_2 = 0 \tag{20}$$

$$\lim_{\Delta t\to 0}\frac{1}{\Delta t}\Pi\left(\overline{\theta}_2,t+\Delta t|\overline{\theta}_1,t\right)=W\left(\overline{\theta}_2|\overline{\theta}_1,t\right) \tag{21}$$

$$\lim_{\Delta t\to 0}\frac{1}{\Delta t}\int_{|\overline{\theta}_2-\overline{\theta}_1|<\varepsilon}\left(\overline{\theta}_{2,i}-\overline{\theta}_{1,i}\right)\Pi\left(\overline{\theta}_2,t+\Delta t|\overline{\theta}_1,t\right)d\overline{\theta}_2=A_i\left(\overline{\theta}_1,t\right)+\mathcal{O}\left(\varepsilon\right) \tag{22}$$

$$\lim_{\Delta t\to 0}\frac{1}{\Delta t}\int_{|\overline{\theta}_2-\overline{\theta}_1|<\varepsilon}\left(\overline{\theta}_{2,i}-\overline{\theta}_{1,i}\right)\left(\overline{\theta}_{2,j}-\overline{\theta}_{1,j}\right)\Pi\left(\overline{\theta}_2,t+\Delta t|\overline{\theta}_1,t\right)d\overline{\theta}_2=B_{i,j}\left(\overline{\theta}_1,t\right)+\mathcal{O}\left(\varepsilon\right) \tag{23}$$

The first condition guarantees the continuity of $\Pi\left(\overline{\theta},t|\overline{\theta}_0,t_0\right)$, while the other three force the finite existence of three parameters. Equation 25 can be furthermore decomposed into the continuous and discrete components:

$$\begin{cases}\Pi\left(\overline{\theta}_3^c,t_3|\overline{\theta}_1^c,t_1\right)=\int\Pi\left(\overline{\theta}_2^c,t_2|\overline{\theta}_1^c,t_1\right)\Pi\left(\overline{\theta}_3^c,t_3|\overline{\theta}_2^c,t_2\right)d\overline{\theta}_2^c\\ \Pi\left(\overline{\theta}_3^d,t_3|\overline{\theta}_1^d,t_1\right)=\int\Pi\left(\overline{\theta}_2^d,t_2|\overline{\theta}_1^d,t_1\right)\Pi\left(\overline{\theta}_3^d,t_3|\overline{\theta}_2^d,t_2\right)d\overline{\theta}_2^d\end{cases}\quad where\ t_1<t_2<t_3 \tag{24}$$

and its differential form is as follows:

$$\begin{cases}\frac{\partial\Pi\left(\overline{\theta}^c,t|\overline{\theta}_0^c,t_0\right)}{\partial t}=\mathcal{L}_{CK}^c\left(\Pi\left(\overline{\theta}^c,t|\overline{\theta}_0^c,t_0\right),\overline{\theta}^d,\overline{\alpha}_{brow},\overline{\alpha}_{staz},\overline{\alpha}_{DS},t\right)\\ \frac{\partial\Pi\left(\overline{\theta}^d,t|\overline{\theta}_0^d,t_0\right)}{\partial t}=\mathcal{L}_{CK}^d\left(\Pi\left(\overline{\theta}^d,t|\overline{\theta}_0^d,t_0\right),\overline{\theta}^c,t\right)\end{cases} \tag{25}$$

where:

- $\Pi\left(\overline{\theta}^c,t|\overline{\theta}_0^c,t_0\right)$ of the system to be in state $\overline{\theta}^c$ at time t given that the system was in $\overline{\theta}_0^c$ at time t_0;
- $\Pi\left(\overline{\theta}^d,t|\overline{\theta}_0^d,t_0\right)$ of the system to be in state $\overline{\theta}^d$ at time t given that the system was in $\overline{\theta}_0^d$ at time t_0;
- $\mathcal{L}_{CK}^c\left(\cdot\right)$ and $\mathcal{L}_{CK}^d\left(\cdot\right)$ are specific Chapman- Kolmogorov operators that will be described in the following section.

2.4 The Chapman-Kolmogorov Equation

The system of equations 2, written in integral form, can be solved in a differential form through the Chapman-Kolmogorov (C-K) operator [7]:

$$\begin{array}{c}\frac{\partial \Pi\left(\overline{\theta}^c,t|\overline{\theta}_0^c,t_0\right)}{\partial t}=-\sum_i \frac{\partial}{\partial \overline{\theta}_i^c}\left(A_i\left(\overline{\theta}^c,\overline{\theta}^d,t\right)\Pi\left(\overline{\theta}^c,t|\overline{\theta}_0^c,t_0\right)\right)+\\ +\frac{1}{2}\sum_{i,j}\frac{\partial^2}{\partial \overline{\theta}_i^c\partial \overline{\theta}_j^c}\left(B_{i,j}\left(\overline{\theta}^c,\overline{\theta}^d,t\right)\Pi\left(\overline{\theta}^c,t|\overline{\theta}_0^c,t_0\right)\right)+\\ +\int\left(W\left(\overline{\theta}^c|\overline{\theta}^{'c},\overline{\theta}^d,t\right)\Pi\left(\overline{\theta}^{'c},t|\overline{\theta}_0^c,t_0\right)-W\left(\overline{\theta}^{'c}|\overline{\theta}^c,\overline{\theta}^d,t\right)\Pi\left(\overline{\theta}^c,t|\overline{\theta}_0^c,t_0\right)\right)d\overline{\theta}^{'c}\end{array} \tag{26}$$

$$\frac{\partial \Pi\left(\overline{\theta}^d,t|\overline{\theta}_0^d,t_0\right)}{\partial t}=\sum_i W\left(\overline{\theta}^d|\overline{\theta}_i^d,\overline{\theta}^c,t\right)\Pi\left(\overline{\theta}_i^d,t|\overline{\theta}^d,t_0\right)-W\left(\overline{\theta}_i^d|\overline{\theta}^d,\overline{\theta}^c,t\right)\Pi\left(\overline{\theta}^d,t|\overline{\theta}_0^d,t_0\right) \tag{27}$$

where:

$$\begin{array}{c}A_i\left(\overline{\theta},t\right)=\begin{cases}0 & if\ \ \overline{\theta}_i\in\overline{\theta}^d\\ f\left(\overline{\theta}^c,\overline{\theta}^d,\alpha_{staz},\alpha_{brow},t\right)+\frac{1}{2}\frac{\partial b\left(\overline{\theta}^c,t\right)}{\partial \overline{\theta}^c}Qb\left(\overline{\theta}^c,t\right) & if\ \ \overline{\theta}_i\notin\overline{\theta}^d\end{cases}\\ B_{i,j}\left(\overline{\theta},t\right)=\begin{cases}0 & if\ \ \overline{\theta}_i\ \ or\ \ \overline{\theta}_j\in\overline{\theta}^d\\ b\left(\overline{\theta}^c,t\right)Qb^T\left(\overline{\theta}^c,t\right) & if\ \ \overline{\theta}_i\ \ or\ \ \overline{\theta}_j\notin\overline{\theta}^d\end{cases}\end{array} \tag{28}$$

This system of equations is composed of four main terms that identify four different types of processes:

- Drift process
- Diffusion process
- Jumps in continuous space
- Jumps in discrete space (component state transitions).

These four processes are described in the following sub-sections.

2.4.1 Drift Process

The drift process is defined by the Lioville's equation:

$$\frac{\partial\,\Pi\left(\overline{\theta}^c,t|\overline{\theta}_0^c,t_0\right)}{\partial t}=\sum_i\frac{\partial}{\partial\overline{\theta}_i^c}\left(A_i\left(\overline{\theta}^c,\overline{\theta}^d,t\right)\Pi\left(\overline{\theta}^c,t|\overline{\theta}_0^c,t_0\right)\right) \tag{29}$$

It is important to note that this equation describes a completely deterministic motion, indicated by the equation:

$$\frac{\partial\,\overline{\theta}^c\left(t\right)}{\partial t}=A_i\left(\overline{\theta}^c,\overline{\theta}^d,t\right) \tag{30}$$

If $\overline{\theta}^c\left(\overline{\theta}_0^c, \overline{\theta}^d, t\right)$ is the solution of Equation 30, then then the solution of the Lioville's equation is:

$$\Pi\left(\overline{\theta}^c, t|\overline{\theta}_0^c, t_0\right) = \delta\left(\overline{\theta}^c - \overline{\theta}^c\left(\overline{\theta}_0^c, \overline{\theta}^d, t\right)\right) \tag{31}$$

provided the initial condition:

$$\Pi\left(\overline{\theta}^c, t|\overline{\theta}_0^c, t_0\right) = \delta\left(\overline{\theta}^c - \overline{\theta}_0^c\right) \tag{32}$$

2.4.2 Diffusion Process

This process is described by the Fokker-Plank equation:

$$\begin{array}{c}\frac{\partial\,\Pi\left(\overline{\theta}^c, t|\overline{\theta}_0^c, t_0\right)}{\partial t} = \sum_i \frac{\partial}{\partial\overline{\theta}_i^c}\left(A_i\left(\overline{\theta}^c, \overline{\theta}^d, t\right)\Pi\left(\overline{\theta}^c, t|\overline{\theta}_0^c, t_0\right)\right) + \\ + \frac{1}{2}\sum_{i,j}\frac{\partial^2}{\partial\overline{\theta}_i^c\partial\overline{\theta}_j^c}\left(B_{i,j}\left(\overline{\theta}^c, \overline{\theta}^d, t\right)\Pi\left(\overline{\theta}^c, t|\overline{\theta}_0^c, t_0\right)\right)\end{array} \tag{33}$$

where $A_i\left(\overline{\theta}^c, \overline{\theta}^d, t\right)$ is the drift vector and $B_{i,j}\left(\overline{\theta}^c, \overline{\theta}^d, t\right)$ is the diffusion matrix.
Provided the initial condition in Equation 32, the Fokker-Plank equation describes a system moving with drift whose velocity is $A\left(\overline{\theta}^c, \overline{\theta}^d, t\right)$ on which is imposed a Gaussian fluctuation with covariance matrix $B\left(\overline{\theta}^c, \overline{\theta}^d, t\right)$.

2.4.3 Jumps in Continuous Space

This process is described by the Master equation:

$$\frac{\partial\,\Pi\left(\overline{\theta}^c, t|\overline{\theta}_0^c, t_0\right)}{\partial t} = \int\left(W\left(\overline{\theta}^c|\overline{\theta}^{'c}, \overline{\theta}^d, t\right)\Pi\left(\overline{\theta}^{'c}, t|\overline{\theta}_0^c, t_0\right) - W\left(\overline{\theta}^{'c}|\overline{\theta}^c, \overline{\theta}^d, t\right)\Pi\left(\overline{\theta}^c, t|\overline{\theta}_0^c, t_0\right)\right)d\overline{\theta}^{'c} \tag{34}$$

Provided the initial condition in Equation 32, it describes a process characterized by straight lines interspersed with discontinuous jumps whose distribution is given by $W\left(\overline{\theta}^c|\overline{\theta}^{'c}, \overline{\theta}^d, t\right)$

2.4.4 Jumps in Discrete Space

Transitions in the discrete space can occur in terms of jumps, then the formulation of

$$\frac{\partial\,\Pi\left(\overline{\theta}^d, t|\overline{\theta}_0^d, t_0\right)}{\partial t} = \mathcal{L}_{CK}^d\left(\Pi\left(\overline{\theta}^d, t|\overline{\theta}_0^d, t_0\right)\right) \tag{35}$$

is similar to the Master equation, recasted for a discrete phase space:

$$\frac{\partial \Pi\left(\overline{\theta}^d, t|\overline{\theta}_0^d, t_0\right)}{\partial t} = \sum_i \left(W\left(\overline{\theta}^d|\overline{\theta}_i^d, \overline{\theta}^c, t\right) \Pi\left(\overline{\theta}_i^d, t|\overline{\theta}_0^d, t_0\right) - W\left(\overline{\theta}_i^d|\overline{\theta}^d, \overline{\theta}^c, t\right) \Pi\left(\overline{\theta}^d, t|\overline{\theta}_0^d, t_0\right)\right) \tag{36}$$

3 Forward Sampling Strategies

In order to perform UQ and dynamic probabilistic risk assessment (DPRA), a sampling strategy needs to be employed. The sampling strategy perturbs the input space (domain of the uncertainties) to explore the response of a complex system in relation to selected figure of merits (FOMs).

The most widely used strategies to perform UQ and PRA are generally collected in RAVEN as ***Forward*** samplers. ***Forward*** samplers include all the strategies that simply perform the sampling of the input space. These strategies sample without exploiting, through learning approaches, the information made available from the outcomes of evaluation previously performed (adaptive sampling) and the common system evolution (patterns) that different sampled calculations can generate in the phase space (Dynamic Event Tree).

RAVEN has several different ***Forward*** samplers:

- *Monte-Carlo*
- *Grid-based*
- *Stratified* and its specialization named *Latin Hyper Cube*.

In addition, RAVEN posses advanced ***Forward*** sampling strategies that:

- Build a grid in the input space selecting evaluation points based on characteristic quadratures as part of stochastic collocation for generalized polynomial chaos method (*Sparse Grid Collocation* sampler);
- Use high-density model reduction (HDMR) a.k.a. Sobol decomposition to approximate a function as the sum of interactions with increasing complexity (*Sobol* sampler).

In the following subsections, the theory behind these sampling methodologies is explained.

3.1 Monte-Carlo

The Monte-Carlo method is one of the most-used methodologies in several mathematic disciplines. It approximates an expectation by the sample mean of a function of simulated random variables. It is based on the laws of large numbers in order to approximate expectations. In order words, it approximates the average response of multiple FOMs relying on multiple random sampling of the input space.
Consider a random variable (eventually multidimensional) X having probability mass function or

probability density function $pdf_X(x)$, which is greater than zero on a set of values χ. Then the expected value of a function f of X is as follows:

$$\begin{aligned} \mathbb{E}(f(X)) &= \textstyle\sum_{x\in\chi} f(x)pdf_X(x) \qquad \text{if} X\ discrete \\ \mathbb{E}(f(X)) &= \textstyle\int_{x\in\chi} f(x)pdf_X(x) \quad \text{if} X\ continuous \end{aligned} \tag{37}$$

Now consider n samples of X, $(x_1, ..., x_n)$, and compute the mean of $f(x)$ over the samples. This computation represents the Monte-Carlo estimate:

$$\mathbb{E}(f(X)) \approx \widetilde{f}_n(x) = \frac{1}{n}\sum_{i=1}^{n} f(x_i) \tag{38}$$

If $\mathbb{E}(f(X))$ exists, then the law of large numbers determines that for any arbitrarily small ε:

$$\lim_{n\to\infty} P(\left|\widetilde{f}_n(X) - \mathbb{E}(f(X))\right| \geq \varepsilon) = 0 \tag{39}$$

The above equation suggests that as n gets larger, then the probability that $\widetilde{f}_n(X)$ deviates from the $\mathbb{E}(f(X))$ becomes smaller. In other words, the more samples are spooned, the more closer the Monte-Carlo estimate of X gets to the real value.
In addition, $\widetilde{f}_n(X)$ represent an unbiased estimate for $\mathbb{E}(f(X))$:

$$\mathbb{E}(\widetilde{f}_n(X)) = \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n} f(x_i)\right) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}(f(x_i)) = \mathbb{E}(f(X)) \tag{40}$$

3.2 Grid

The Grid sampling method (also known as Full Factorial Design of Experiment) represents one of the simplest methodologies that can be employed in order to explore the interaction of multiple random variables with respect selected FOMs. The goal of the Grid-based sampling strategy is to explore the interaction of multiple random variables (i.e.,uncertainties) with respect to selected FOMs. Indeed, this method is mainly used to perform parametric analysis of the system response rather than a probabilistic one. This method discretizes the domain of the uncertainties in a user-defined number of intervals (see Figure 1) and record the response of the model (e.g. a system code) at each coordinate (i.e., combination of the uncertainties) of the grid.
This method starts from the assumption that each coordinate on the grid is representative, with respect to the FOMs of interest, of the surrounding grid cell. In other words, it is assumed that the response of a system does not significantly change within the hyper-volume surrounding each grid coordinate (red square in Figure 1).

Similarly to what has been already reported for the Monte-Carlo sampling, consider a random variable X having PDF $pdf_X(x)$ and, consequentially, CDF $cdf_X(x)$ in the domain χ. Then the

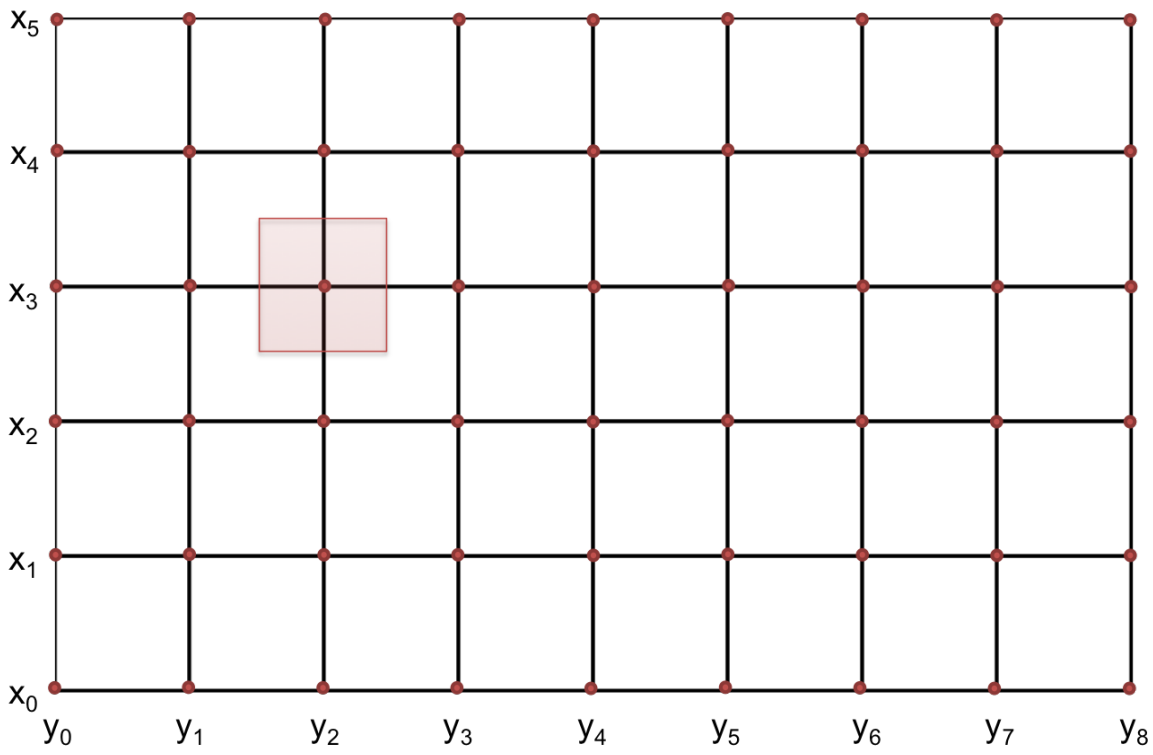

Figure 1. Example of 2-Dimensional grid discretization.

expected value of a function f of X is as follows:

$$\begin{aligned} \mathbb{E}(f(X)) &= \textstyle\sum_{x\in\chi} f(x) pdf_X(x) \qquad if X\ discrete \\ \mathbb{E}(f(X)) &= \textstyle\int_{x\in\chi} f(x) pdf_X(x) \qquad if X\ continuous \end{aligned} \tag{41}$$

In the Grid approach, the domain of X is discretized in a finite number of intervals. Recall that each node of this discretization is representative of the surrounding hyper-volume. This means that a weight needs to be associated with each coordinate of the resulting grid:

$$w_i = cdf_X(x_{i+1/2}) - cdf_X(x_{i-1/2}) \tag{42}$$

Now consider a $n-$discretization of the domain of X, $(x_1, ..., x_n)$ and compute the mean of $f(x)$ over the discretization. Based on the previous equation, the computation of the expected value of $f(x)$ is as follows:

$$\mathbb{E}(f(X)) \approx \widetilde{f}_n(x) = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} f(x_i) \times w_i \tag{43}$$

If the number of uncertainties under consideration is greater than one (m), the above equation

becomes:

$$\mathbb{E}(f(\overline{X})) \approx \widetilde{f}_n(\overline{x}) = \frac{1}{\sum_{i=1}^{n}\prod_{j=1}^{m} w_{i,j}} \sum_{i=1}^{n} f(\overline{x}_i) \times \prod_{j=1}^{m} w_{i,j} \tag{44}$$

3.3 Stratified

The Stratified sampling is a class of methods that relies on the assumption that the input space (i.e.,uncertainties) can be separated in regions (strata) based on similarity of the response of the system for input set within the same strata. Following this assumption, the most rewarding (in terms of computational cost vs. knowledge gain) sampling strategy would be to place one sample for each region. In this way, the same information is not collected more than once and all the prototypical behavior are sampled at least once. In Figure 2, the Stratified sampling approach is exemplified.

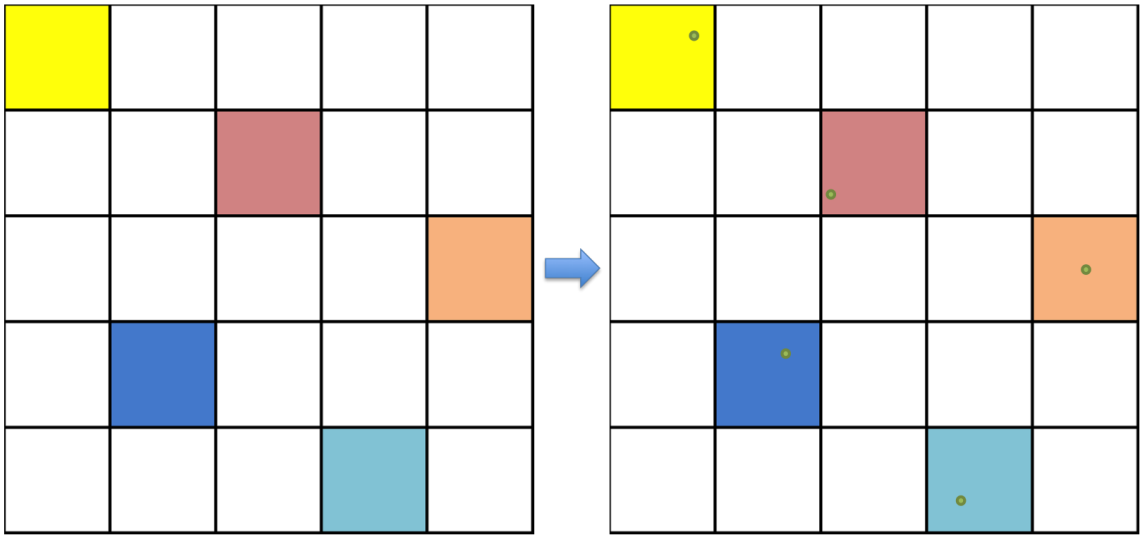

Figure 2. Example of Stratified sampling approach.

The Stratified sampling approach is a method for the exploration of the input space that consists of dividing the uncertain domain into subgroups before sampling. In the Stratified sampling, these subgroups must be:

- Mutually exclusive: every element in the population must be assigned to only one stratum (subgroup)
- Collectively exhaustive: no population element can be excluded.

Then, simple random sampling or systematic sampling is applied within each stratum. The Latin Hyper-Cube sampling represents a specialized version of the stratified approach, when the domain strata are constructed in equally-probable CDF bins.
Similarly to what has been already reported for the Grid sampling, consider a set of m random variables $X_j,\ j = 1, .., m$ having PDFs $pdf_{X_j}(x_j)$ and, consequentially, CDF $cdf_{X_j}(x_j)$ in the domain χ_j. Then the expected value of a function f of $X_j,\ j = 1, .., m$ is as follows:

$$\begin{aligned} \mathbb{E}(f(\overline{X})) &= \sum f(x) \prod_{j=1}^{m} pdf_{X_j}(x_j) \qquad if \overline{X}\ discrete \\ \mathbb{E}(f(\overline{X})) &= \int f(x) \prod_{j=1}^{m} pdf_{X_j}(x_j) \quad if \overline{X}\ continuous \end{aligned} \tag{45}$$

In the Stratified approach, the domain of $\overline{X}$ is discretized in a set of strata. Consequentially, similarly to the Grid sampling, a weight needs to be associated with each realization in the input space:

$$w_i = \frac{\prod_{j=1}^{m} \left[cdf_{X_j}(x_{i,j+1}) - cdf_{X_j}(x_{i,j}) \right]}{\sum_{points} \prod_{j=1}^{m} \left[cdf_{X_j}(x_{i,j+1}) - cdf_{X_j}(x_{i,j}) \right]} \tag{46}$$

Each realization carries a weight representative of each stratum.
Now consider an $n-$strata of the domain of $\overline{X}$, and compute the expected value of $f(x)$ over the discretization. Based on the previous equation, the computation of the expected value of $f(x)$ is as follows:

$$\mathbb{E}(f(\overline{X})) \approx \widetilde{f}_n(\overline{x}) = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} f(\overline{x}_i) \times w_i \tag{47}$$

3.4 Sparse Grid Collocation

The Sparse Grid Collocation sampler represents an advanced methodology to perform Uncertainty Quantification. They aim to explore the input space leveraging the information contained in the associated probability density functions. It builds on generic Grid sampling by selecting evaluation points based on characteristic quadratures as part of stochastic collocation for generalized polynomial chaos uncertainty quantification. In collocation an N-D grid is constructed, with each uncertain variable providing an axis. Along each axis, the points of evaluation correspond to quadrature points necessary to integrate polynomials. In the simplest (and most naive) case, a N-D tensor product of all possible combinations of points from each dimension's quadrature is constructed as sampling points. The number of necessary samples can be reduced by employing Smolyak-like sparse grid algorithms, which use reduced combinations of polynomial orders to reduce the necessary sampling space.

3.4.1 Generalized Polynomial Chaos

In general, polynomial chaos expansion (PCE) methods seek to interpolate the simulation code as a combination of polynomials of varying degree in each dimension of the input space. Originally

Wiener proposed expanding in Hermite polynomials for Gaussian-normal distributed variables [8]. Askey and Wilson generalized Hermite polynomials to include Jacobi polynomials, including Legendre and Laguerre polynomials [9]. Xiu and Karniadakis combines these concepts to perform PCE for a range of Gaussian-based distributions with corresponding polynomials, including Legendre polynomials for uniform distributions, Laguerre polynomials for Gamma distributions, and Jacobi polynomials for Beta distributions [10].

In each of these cases, a probability-weighted integral over the distribution can be cast in a way that the corresponding polynomials are orthogonal over the same weight and interval. These chaos Wiener-Askey polynomials were used by Xiu and Karniadakis to develop the generalized polynomial chaos expansion method (gPC), including a transformation for applying the same method to arbitrary distributions (as long as they have a known inverse CDF) [10]. Two significant methodologies have grown from gPC application. The first makes use of Lagrange polynomials to expand the original function or simulation code, as they can be made orthogonal over the same domain as the distributions [11]; the other uses the Wiener-Askey polynomials [10].

Consider a simulation code that produces a quantity of interest u as a function $u(Y)$ whose arguments are the uncertain, distributed input parameters $Y = (Y_1, \ldots, Y_n, \ldots, Y_N)$. A particular realization ω of Y_n is expressed by $Y_n(\omega)$, and a single realization of the entire input space results in a solution to the function as $u(Y(\omega))$. Obtaining a realization of $u(Y)$ may take considerable computation time and effort. $u(Y)$ gets expanded in orthonormal multidimensional polynomials $\Phi_k(Y)$, where k is a multi-index tracking the polynomial order in each axis of the polynomial Hilbert space, and $\Phi_k(Y)$ is constructed as

$$\Phi_k(Y) = \prod_{n=1}^{N} \phi_{k_n}(Y_n), \tag{48}$$

where $\phi_{k_n}(Y_n)$ is a single-dimension Wiener-Askey orthonormal polynomial of order k_n and $k = (k_1, \ldots, k_n, \ldots, k_N)$, $k_n \in \mathbb{N}^0$. For example, given $u(y_1, y_2, y_3)$, $k = (2, 1, 4)$ is the multi-index of the product of a second-order polynomial in y_1, a first-order polynomial in y_2, and a fourth-order polynomial in y_4. The gPC for $u(Y)$ using this notation is

$$u(Y) \approx \sum_{k \in \Lambda(L)} u_k \Phi_k(Y), \tag{49}$$

where u_k is a scalar weighting polynomial coefficient. The polynomials used in the expansion are determined by the set of multi-indices $\Lambda(L)$, where L is a truncation order. In the limit that Λ contains all possible combinations of polynomials of any order, Eq. 48 is exact. Practically, however, Λ is truncated to some finite set of combinations, discussed in section 3.4.2.

Using the orthonormal properties of the Wiener-Askey polynomials,

$$\int_{\Omega} \Phi_k(Y) \Phi_{\hat{k}}(Y) \rho(Y) dY = \delta_{k\hat{k}}, \tag{50}$$

where $\rho(Y)$ is the combined PDF of Y, Ω is the multidimensional domain of Y, and δ_{nm} is the Dirac delta, an expression of the polynomial expansion coefficients can be isolated. We multiply both sides of Eq. 48 by $\Phi_{\hat{k}}(Y)$, integrate both sides over the probability-weighted input domain, and sum over all $\hat{k}$ to obtain the coefficients, sometimes referred to as polynomial expansion moments,

$$u_k = \frac{\langle u(Y)\Phi_k(Y)\rangle}{\langle \Phi_k(Y)^2\rangle}, \tag{51}$$

$$= \langle u(Y)\Phi_k(Y)\rangle, \tag{52}$$

where we use the angled bracket notation to denote the probability-weighted inner product,

$$\langle f(Y)\rangle \equiv \int_\Omega f(Y)\rho(Y)dY. \tag{53}$$

When $u(Y)$ has an analytic form, these coefficients can be solved by integration; however, in general other methods must be applied to numerically perform the integral. While tools such as Monte Carlo integration can be used to evaluate the integral, the properties of Gaussian quadratures, because of the probability weights and domain, can be harnessed. This stochastic collocation method is discussed in section 3.4.4.

3.4.2 Polynomial Index Set Construction

The main concern in expanding a function in interpolating multidimensional polynomials is choosing appropriate polynomials to make up the expansion. There are many generic ways by which a polynomial set can be constructed. Here three static approaches are presented:

- Tensor Product
- Total Degree
- Hyperbolic Cross.

In the nominal tensor product case, $\Lambda(L)$ contains all possible combinations of polynomial indices up to truncation order L in each dimension, as:

$$\Lambda_{\text{TP}}(L) = \left\{\bar{p} = (p_1, \cdots, p_N) : \max_{1\leq n\leq N} p_n \leq L\right\}. \tag{54}$$

The cardinality of this index set is $|\Lambda_{\text{TP}}(L)| = (L+1)^N$. For example, for a two-dimensional input space (N=2) and truncation limit $L = 3$, the index set $\Lambda_{\text{TP}}(3)$ is given in Table 1, where the notation $(1, 2)$ signifies the product of a polynomial that is first order in Y_1 and second order in Y_2.

(3,0)	(3,1)	(3,2)	(3,3)
(2,0)	(2,1)	(2,2)	(2,3)
(1,0)	(1,1)	(1,2)	(1,3)
(0,0)	(0,1)	(0,2)	(0,3)

Table 1. Tensor Product Index Set, $N = 2, L = 3$.

It is evident there is some inefficiencies in this index set. First, it suffers dramatically from the *curse of dimensionality*; that is, the number of polynomials required grows exponentially with increasing dimensions. Second, the total order of polynomials is not considered. Assuming the contribution of each higher-order polynomial is smaller than lower-order polynomials, the (3,3) term is contributing sixth-order corrections that are likely smaller than the error introduced by ignoring fourth-order corrections (4,0) and (0,4). This leads to the development of the *total degree* (TD) and *hyperbolic cross* (HC) polynomial index set construction strategies [12].

In TD, only multidimensional polynomials whose *total* order at most L are permitted,

$$\Lambda_{\text{TD}}(L) = \Big\{\bar{p} = (p_1, \cdots, p_N) : \sum_{n=1}^{N} p_n \leq L\Big\}. \tag{55}$$

The cardinality of this index set is $|\Lambda_{\text{TD}}(L)| = \binom{L+N}{N}$, which grows with increasing dimensions much more slowly than TP. For the same $N = 2, L = 3$ case above, the TD index set is given in Table 2.

(3,0)			
(2,0)	(2,1)		
(1,0)	(1,1)	(1,2)	
(0,0)	(0,1)	(0,2)	(0,3)

Table 2. Total Degree Index Set, $N = 2, L = 3$

In HC, the *product* of polynomial orders is used to restrict allowed polynomials in the index set. This tends to polarize the expansion, emphasizing higher-order polynomials in each dimension but lower-order polynomials in combinations of dimensions, as:

$$\Lambda_{\text{HC}}(L) = \Big\{\bar{p} = (p_1, \ldots, p_N) : \prod_{n=1}^{N} p_n + 1 \leq L + 1\Big\}. \tag{56}$$

The cardinality of this index set is bounded by $|\Lambda_{\text{HC}}(L)| \leq (L+1)(1+\log(L+1))^{N-1}$. It grows even more slowly than TD with increasing dimension, as shown in Table 3 for $N = 2, L = 3$.

(3,0)			
(2,0)			
(1,0)	(1,1)		
(0,0)	(0,1)	(0,2)	(0,3)

Table 3. Hyperbolic Cross Index Set, $N = 2, L = 3$

It has been shown that the effectiveness of TD and HC as index set choices depends strongly on the regularity of the responce [12]. TD tends to be most effective for infinitely-continuous response surfaces, while HC is more effective for surfaces with limited smoothness or discontinuities.

3.4.3 Anisotropy

While using TD or HC to construct the polynomial index set combats the curse of dimensionality present in TP, it is not eliminated and continues to be an issue for problems of large dimensionality. Another method that can be applied to mitigate this issue is index set anisotropy, or the unequal treatment of various dimensions. In this strategy, weighting factors $\alpha = (\alpha_1, \ldots, \alpha_n, \ldots, \alpha_N)$ are applied in each dimension to allow additional polynomials in some dimensions and less in others. This change adjusts the TD and HC construction rules as follows, where $|\alpha|_1$ is the one-norm of α.

$$\tilde{\Lambda}_{TD}(L) = \Big\{\bar{p} = (p_1, \ldots, p_N) : \sum_{n=1}^{N} \alpha_n p_n \leq |\boldsymbol{\alpha}|_1 L\Big\}, \tag{57}$$

$$\tilde{\Lambda}_{\text{HC}}(L) = \Big\{\bar{p} = (p_1, \cdots, p_N) : \prod_{n=1}^{N} (p_n + 1)^{\alpha_n} \leq (L+1)^{|\boldsymbol{\alpha}|_1}\Big\} \tag{58}$$

As it is desirable to obtain the isotropic case from a reduction of the anisotropic cases, define the one-norm for the weights is defined as:

$$|\alpha|_1 = \frac{\sum_{n=1}^{N} \alpha_n}{N}. \tag{59}$$

Considering the same case above ($N = 2, L = 3$), it can be applied weights $\alpha_1 = 5, \alpha_2 = 3$, and the resulting index sets are Tables 4 (TD) and 5 (HC).

There are many methods by which anisotropy weights can be assigned. Often, if a problem is well-known to an analyst, it may be enough to use heuristics to assign importance arbitrarily. Otherwise, a smaller uncertainty quantification solve can be used to roughly determine sensitivity coefficients (such as Pearson coefficients), and the inverse of those can then be applied as anisotropy weights. Sobol coefficients obtained from first- or second-order HDMR, an additional

(2,0)				
(1,0)	(1,1)	(1,2)		
(0,0)	(0,1)	(0,2)	(0,3)	(0,4)

Table 4. Anisotropic Total Degree Index Set, $N = 2, L = 3$.

(1,0)			
(0,0)	(0,1)	(0,2)	(0,3)

Table 5. Anisotropic Hyperbolic Cross Index Set, $N = 2, L = 3$.

sampling strategy present in RAVEN, could also serve as a basis for these weights. A good choice of anisotropy weight can greatly speed up convergence; however, a poor choice can slow convergence considerably, as computational resources are used to resolve low-importance dimensions.

3.4.4 Stochastic Collocation

Stochastic collocation is the process of using collocated points to approximate integrals of stochastic space numerically. In particular consider using Gaussian quadratures (Legendre, Hermite, Laguerre, and Jacobi) corresponding to the polynomial expansion polynomials for numerical integration. Quadrature integration takes the form:

$$\int_a^b f(x)\rho(x) = \sum_{\ell=1}^{\infty} w_\ell f(x_\ell), \tag{60}$$

$$\approx \sum_{\ell=1}^{\hat{L}} w_\ell f(x_\ell), \tag{61}$$

where w_ℓ, x_ℓ are corresponding points and weights belonging to the quadrature set, truncated at order $\hat{L}$. At this point, this $\hat{L}$ should not be confused with the polynomial expansion truncation order L. This expression can be simplified using the operator notation

$$q^{(\hat{L})}[f(x)] \equiv \sum_{\ell=1}^{\hat{L}} w_\ell f(x_\ell). \tag{62}$$

A nominal multidimensional quadrature is the tensor product of individual quadrature weights and points, and can be written

$$Q^{(\boldsymbol{L})} = q_1^{(\hat{L}_1)} \otimes q_2^{(\hat{L}_2)} \otimes \cdots, \tag{63}$$

$$= \bigotimes_{n=1}^{N} q_n^{(\hat{L}_n)}. \tag{64}$$

It is worth noting that each quadrature may have distinct points and weights; they need to not be constructed using the same quadrature rule. In general, one-dimensional Gaussian quadrature excels in exactly integrating polynomials of order $2p-1$ using p points and weights; equivalently, it requires $(p+1)/2$ points to integrate an order p polynomial.

For convenience, the coefficient integral to be evaluated is here reported again, Eq. 51.

$$u_k = \langle u(Y)\Phi_k(Y)\rangle. \tag{65}$$

This integral can be approximated with the appropriate Gaussian quadrature as

$$u_k \approx Q^{(\hat{\boldsymbol{L}})}[u(Y)\Phi_k(Y)], \tag{66}$$

where bold vector notation is used to note the order of each individual quadrature, $\hat{\boldsymbol{L}} = [\hat{L}_1, \ldots, \hat{L}_n, \ldots, \hat{L}_N]$. For clarity, the bold notation is removed and it is assumed a one-dimensional problem, which extrapolates as expected into the multidimensional case.

$$u_k \approx q^{(\hat{L})}[u(Y)\Phi_k(Y)], \tag{67}$$

$$= \sum_{\ell=1}^{\hat{L}} w_\ell u(Y_\ell)\Phi_k(Y_\ell). \tag{68}$$

To determine the quadrature order $\hat{L}$ needed to accurately integrate this expression, consider the gPC formulation for $u(Y)$ in Eq. 48 and replace it in the sum,

$$u_k \approx \sum_{\ell=1}^{\hat{L}} w_\ell \Phi_k(Y_\ell) \sum_{k\in\Lambda(L)} u_{\hat{k}}\Phi_{\hat{k}}(Y_\ell). \tag{69}$$

Using orthogonal properties of the polynomials, this reduces as $\hat{L} \to \infty$ to

$$u_k \approx \sum_{\ell=1}^{\hat{L}} w_\ell u_k \Phi_k(Y_\ell)^2. \tag{70}$$

Thus, the integral, to the same error introduced by truncating the gPC expansion, the quadrature is approximating an integral of order $2k$. As a result, the quadrature order should be ordered:

$$p = \frac{2k+1}{2} = k + \frac{1}{2} < k+1, \tag{71}$$

so it can conservatively used $p = k + 1$. In the case of the largest polynomials with order $k = L$, the quadrature size $\hat{L}$ is the same as $L + 1$. It is worth noting that if $u(Y)$ is effectively of much higher-order polynomial than L, this equality for quadrature order does not hold true; however, it also means that gPC of order L will be a poor approximation.

While a tensor product of highest-necessary quadrature orders could serve as a suitable multidimensional quadrature set, we can make use of Smolyak-like sparse quadratures to reduce the number of function evaluations necessary for the TD and HC polynomial index set construction strategies.

3.4.5 Smolyak Sparse Grids

Smolyak sparse grids [13] are an attempt to discover the smallest necessary quadrature set to integrate a multidimensional integral with varying orders of predetermined quadrature sets. In RAVEN case, the polynomial index sets determine the quadrature orders each one needs in each dimension to be integrated accurately. For example, the polynomial index set point (2,1,3) requires three points in Y_1, two in Y_2, and four in Y_3,or

$$Q^{(2,1,3)} = q_1^{(3)} \otimes q_2^{(2)} \otimes q_3^{(4)}. \tag{72}$$

The full tensor grid of all collocation points would be the tensor product of all quadrature for all points, or

$$Q^{(\Lambda(L))} = \bigotimes_{k \in \Lambda} Q^{(k)}. \tag{73}$$

Smolyak sparse grids consolidate this tensor form by adding together the points from tensor products of subset quadrature sets. Returning momentarily to a one-dimensional problem, introduce the notation

$$\Delta_k^{(\hat{L})}[f(x)] \equiv (q_k^{(\hat{L})} - q_{k-1}^{(\hat{L})})[f(x)], \tag{74}$$

$$q_0^{(\hat{L})}[f(x)] = 0. \tag{75}$$

A Smolyak sparse grid is then defined and applied to the desired integral in Eq. 51,

$$S_{\Lambda,N}^{(\hat{\boldsymbol{L}})}[u(Y)\Phi_k(Y)] = \sum_{k \in \Lambda(L)} \left(\Delta_{k_1}^{(\hat{L}_1)} \otimes \cdots \otimes \Delta_{k_N}^{(\hat{L}_N)} \right) [u(Y)\Phi_k(Y)]. \tag{76}$$

Equivalently, and in a more algorithm-friendly approach,

$$S_{\Lambda,N}^{(\hat{\boldsymbol{L}})}[u(Y)\Phi_k(Y)] = \sum_{k \in \Lambda(L)} c(k) \bigotimes_{n=1}^{N} q_n^{(\hat{L}_n)}[u(Y)\Phi_k(Y)] \tag{77}$$

where

$$c(k) = \sum_{\substack{j=\{0,1\}^N, \\ k+j \in \Lambda}} (-1)^{|j|_1}, \tag{78}$$

using the traditional 1-norm for $|j|_1$. The values for u_k can then be calculated as

$$u_k = \langle u(Y)\Phi_k(Y)\rangle, \tag{79}$$

$$\approx S^{(\hat{\boldsymbol{L}})}_{\Lambda,N}[u(Y)\Phi_k(Y)]. \tag{80}$$

With this numerical method to determine coefficients, a complete method for performing SCgPC analysis in an algorithmic manner is obtained.

4 Adaptive Sampling Strategies

Performing UQ and Dynamic PRA can be challenging from a computational point of view. The *Forward* sampling strategies reported in the previous Section can lead to a large number of unnecessary evaluations of the physical model leading to an unacceptable resource expenses (CPU time). In addition, the *Forward* methodologies are not designed to leverage the information content that is extractable from the simulations already concluded.

To overcome these limitations, in RAVEN several adaptive algorithms are available:

1. *Limit Surface Search*
2. *Adaptive Dynamic Event Tree*
3. *Adaptive Hybrid Dynamic Event Tree*
4. *Adaptive Sparse Grid*
5. *Adaptive Sobol Decomposition*.

In this Section, only the first algorithm is going to be reported.

4.1 Limit Surface Search Method

The motivation behind the choice of adaptive sampling strategies is that numerical 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 unfeasible using finite computing resources. During DPRA 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. The "Limit Surface Search method" is a scheme where few observations, obtained from the model run, are used to build a simpler and computationally faster mathematical representation of the model, ROM, also known as Surrogate Model (ROM or SM). The ROM (see Section **??**) 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 3). The new observations are used to update the ROM and this process iterates until, within a certain metric, it is converged.

In the case of the "Limit Surface (LS) Search method", a ROM is used to determine the location in the input space where further observations are 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.

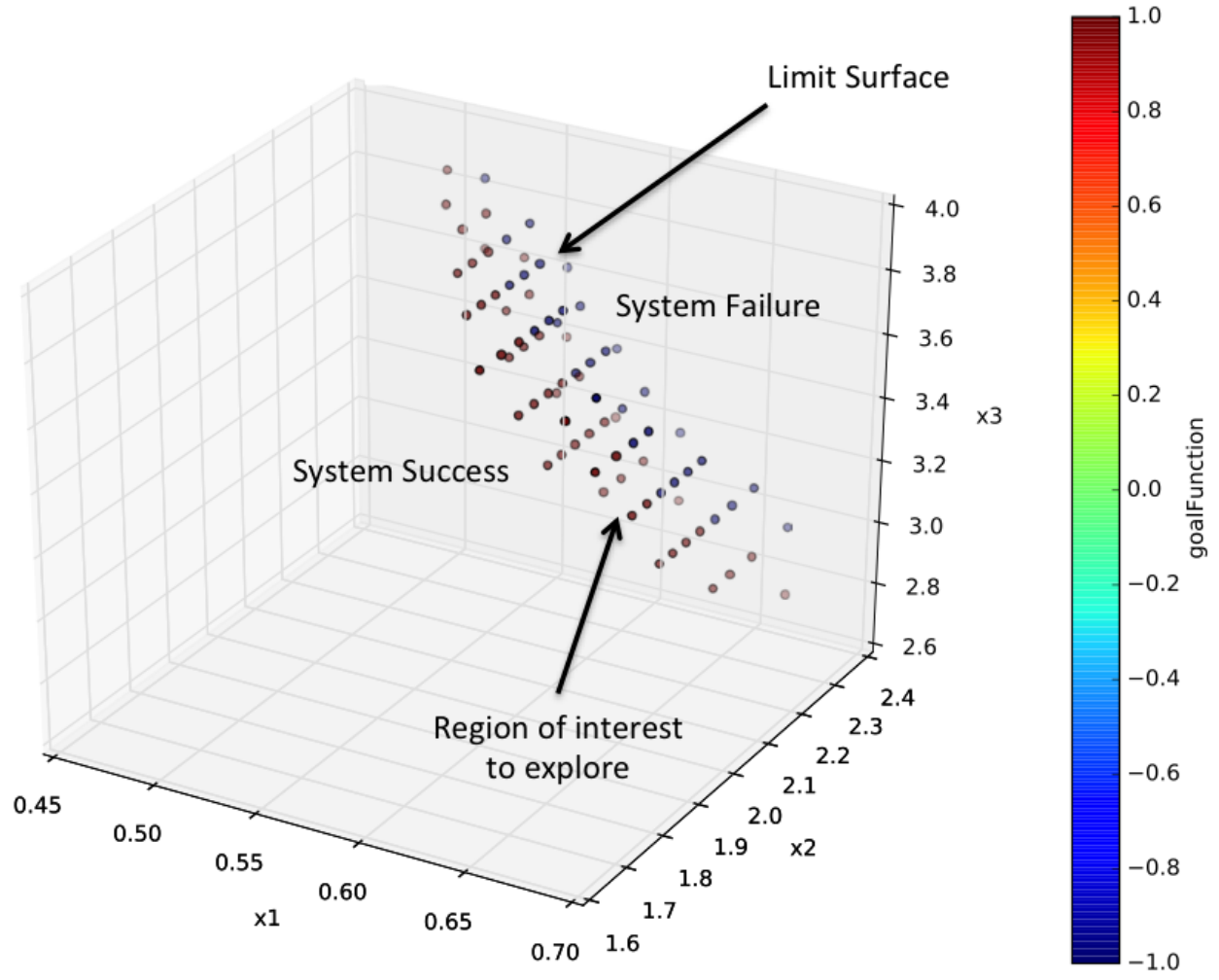

Figure 3. Example of limit surface in the uncertain space.

4.1.1 Limit Surface Theory

To properly explain the LS concept and relative properties, it is necessary to analyze the idea behind 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 the Eq. 1 in Section **??**. The equation can be rewritten as follows:

$$\frac{\partial \overline{\theta}}{\partial t} = \overline{H}\left(\overline{x}, \overline{p}, t\right) \tag{81}$$

where:

- $\overline{\theta}$ represents the coordinate of the system in the phase space

- $(\overline{x}, \overline{p}, t)$ independent variables that are separated, respectively, in spatial, temporal, and parametric space (distinction between $(\overline{x}, \overline{p}, t)$ is purely based on engineering considerations).

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

$$C\left(\overline{\theta}, \overline{x}, \overline{p}, t\right) = C\left(H\left(\overline{x}, \overline{p}, t\right), \overline{x}, \overline{p}, t\right) = C\left(\overline{x}, \overline{p}, t\right) \tag{82}$$

Without loss of generality, lets assume that C does not depend on time (e.g. $C \leftarrow \int_{t_0}^{t_{end}} dt C\left(\overline{x}, \overline{p}, t\right)$):

$$C = C\left(\overline{x}, \overline{p}\right) \tag{83}$$

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) [14], which allows ensuring that all the stochastic phenomena in the system are representable as PDFs in the uncertain domain (see Section **??**),. This allows combining the parametric space with the initial condition space:

$$\begin{array}{c} \left(\overline{x}\right) \leftarrow \left(\overline{x}, \overline{p}\right) \\ C\left(\overline{x}\right) \leftarrow C\left(\overline{x}, \overline{p}\right) \end{array} \tag{84}$$

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. The full heuristic approach to the characterization of system stochastic behaviors is reported in Section **??**.
Under the above simplifications, it is possible to identify the region of the input space (V) leading to a specific outcome of the "goal" function. For example, it can be defined the failure region V_F as the region of the input space where $C = 1$:

$$V_F = \left\{\forall \overline{x} | C\left(\overline{x}\right) = 1\right\} \tag{85}$$

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

$$V_F^c = \left\{\forall \overline{x} | C\left(\overline{x}\right) = 0\right\} \tag{86}$$

Its boundary is the named LS:

$$L_S = \partial V_F^c = \partial\left\{\forall \overline{x} | C\left(\overline{x}\right) = 1\right\} \tag{87}$$

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, in addition, its informative content can be used to analyze the system to characterize its behavior from a stochastic point of view. Consider $\overline{x} \in V$ and $\overline{x} \sim \overline{X}$, where $\overline{x}$ is the random variate realization of the stochastic variable $\overline{X}$. If $pdf_{\overline{X}}(\overline{x})$ is the probability density function of $\overline{X}$, the failure probability of the system (P_F) is:

$$P_F = \int_V d\overline{x} C(\overline{x}) pdf_{\overline{X}}(\overline{x}) = \int_{V_F + V_F^c} d\overline{x} C(\overline{x}) pdf_{\overline{X}}(\overline{x}) \tag{88}$$

And, based on the definition given in Equations 84 and 85:

$$\int_{V_F} d\overline{x} pdf_{\overline{X}}(\overline{x}) \tag{89}$$

Equations 88 and 89 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 (see Figure 4).
It is beneficial for better understanding to assess the LS concept through an example related to the safety of an Nuclear Power Plant (NPP). As an example, consider a station black out (SBO) scenario in an NPP. Suppose that the only uncertain parameters are:

- t_F: Temperature that would determine the failure of the fuel cladding
- 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:

$$t_F \sim pdf_{T_F}(T_F) = \begin{cases} 0 & if\ t_F < t_{F_{min}} \\ \frac{1}{\left(t_{F_{max}} - t_{F_{min}}\right) = \Delta t_F} & \\ 0 & if\ t_F > t_{F_{max}} \end{cases} \tag{90}$$

$$rt_{DGs} \sim pdf_{RT_{DGs}}(rt_{DGs}) = \begin{cases} 0 & if\ rt_{DGs} < rt_{DGs_{min}} \\ \frac{1}{\left(rt_{DGs_{max}} - rt_{DGs_{min}}\right) = \Delta rt_{DGs}} & \\ 0 & if\ rt_{DGs} > rt_{DGs_{max}} \end{cases} \tag{91}$$

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

$$t = t_0 + \alpha \times rt_{DGs}^2 \tag{92}$$

and that the $t_{F_{min}} > t_0 + \alpha \times rt_{DGs_{min}}^2$ and $t_{F_{max}} < t_0 + \alpha \times rt_{DGs_{max}}^2$. The LS, failure region, and active part of the failure region (failure region with non-zero probability) are illustrated, for example, in Figure 4 (in agreement with the above assumptions). In this case, the transition/failure

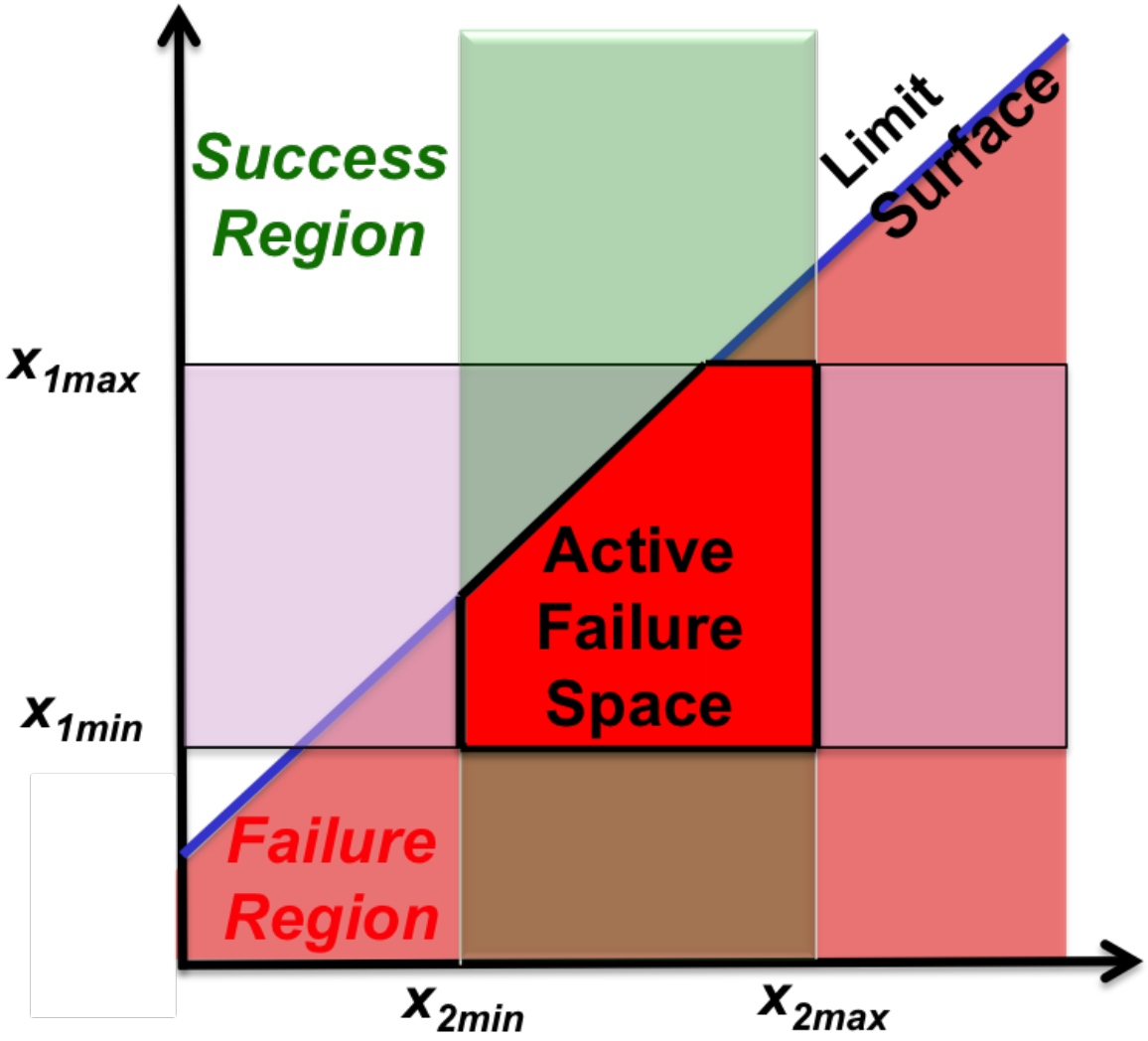

Figure 4. Example of limit surface probability of failure region.

probability is evaluated as follows:

$$P_F = \int_{V_F} d\overline{x}\; pdf_{\overline{X}}\left(\overline{x}\right) = \int_0^{+\infty} d\overline{t_F}\; pdf_{T_F}\left(T_F\right) \int_{\sqrt{\frac{t_F - t_0}{\alpha}}}^{+\infty} d\, rt_{DGs}\; pdf_{RT_{DGs}}\left(rt_{DGs}\right) =$$
$$= \int_{t_{F_{min}}}^{t_{F_{max}}} dt_F \frac{1}{t_{F_{max}} - t_{F_{min}}} \int_{\sqrt{\frac{t_F - t_0}{\alpha}}}^{rt_{DGs_{max}}} d\, rt_{DGs} \frac{1}{rt_{DGs_{max}} - rt_{DGs_{min}}} = \tag{93}$$
$$= \frac{rt_{DGs_{max}}}{\Delta rt_{DGs}} + \frac{2\alpha}{3(\Delta rt_{DGs} \Delta t_F)} \left(\sqrt[3/2]{\frac{t_{F_{min}} - t_0}{\alpha}} - \sqrt[3/2]{\frac{t_{F_{max}} - t_0}{\alpha}} \right)$$

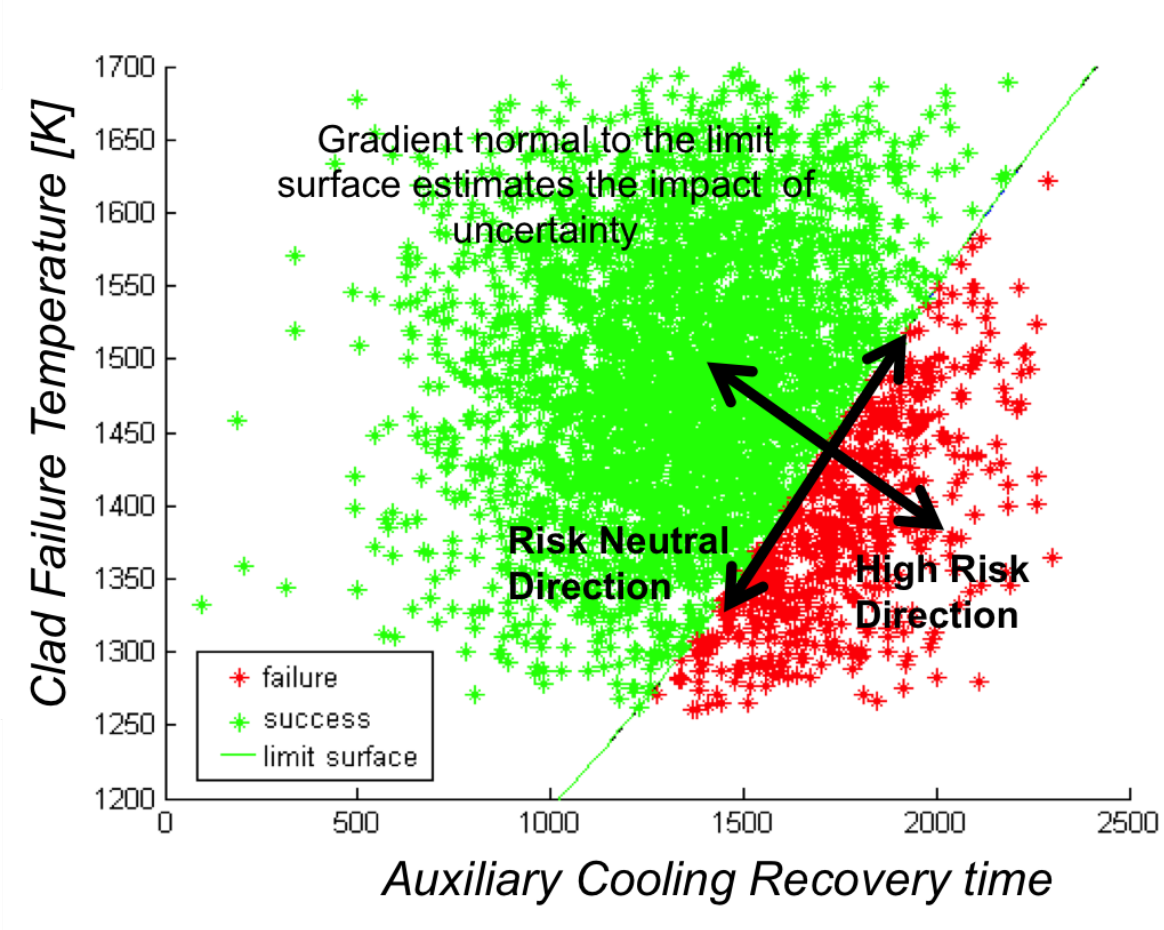

Figure 5. Example of limit surface highlighting the risk directions.

This simple example is useful to understand how the LS is defined in a practical application (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 computed LS is shown in Figure 5. In this figure the neutral and high risk directions are highlighted.

4.1.1.1 Limit Surface Search Algorithm

The identification of the LS location is extremely challenging, depending on the particular physics/phenomena that are investigated. To identify the real location of the LS, the evaluation of system responses is needed, through the high-fidelity code (RELAP 7, RELAP5-3D, etc.), 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 6, for example, green is used for grid nodes with a "goal" function that equals 0 and red when the "goal" function equals 1. Each evaluation of the "goal" function in one of the grid nodes

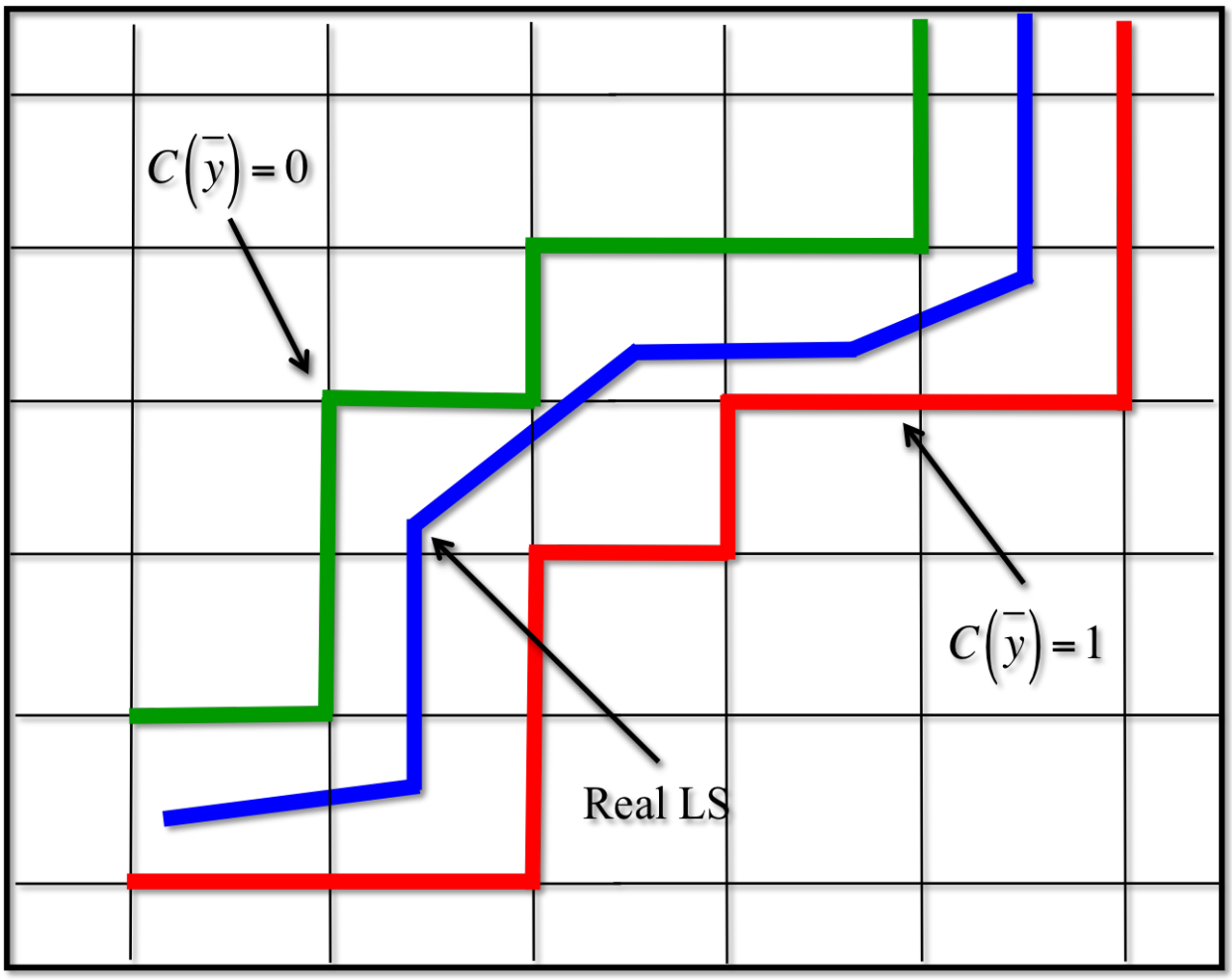

Figure 6. Example of limit surface search evaluation grid (where $\overline{y} = \overline{\theta}$).

implies the evaluation of the high-fidelity code (e.g. system simulator) for the corresponding set of entries in the uncertain space. As already mentioned, the evaluation of the high fidelity code is computationally expensive and, in order to identify the LS, one 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 to be accelerated using "predicting" methods that are represented by the employment of supervised learning algorithms (i.e., ROMs).

This approach 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 $\{\overline{x}_k\}$ are selected via one of the forward sampling strategies (e.g., stratified or Monte Carlo)
2. The high fidelity code is used to compute the status of the system for the set of points in the input set: $\left\{\overline{\theta}(t)\right\}_k = H\left(\left\{\overline{x}\right\}_k, t\right)$.
3. The "goal" function is evaluated at the phase space coordinate of the system: $\{c\}_k = C\left(\left\{\overline{\theta}(t)\right\}_k\right)$
4. The set of pairs $\{(\overline{x}, c)_k\}$ are used to train a ROM of type classifier, $G\left(\{\overline{x}_k\}\right)$
5. The ROM classifier is used to predict the values of the "goal" function for all the N nodes of the N-D grid in the domain space:
$$\left(G\left(\left\{\overline{x}\right\}_j\right) \sim \{c\}_j, j = 1, ..., N\right) \tag{94}$$
6. The values of the "goal" function are used to determine the LS location based on the change of values of $\{c\}_j$:
$$\{c\}_j \rightarrow \partial V_F \tag{95}$$
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 7. Note that there is an additional requirement regarding the LS search algorithm:the LS location has to stay constant for a certain number (user defined) of consecutive iterations. The reason for this choice is determined by the attempt to

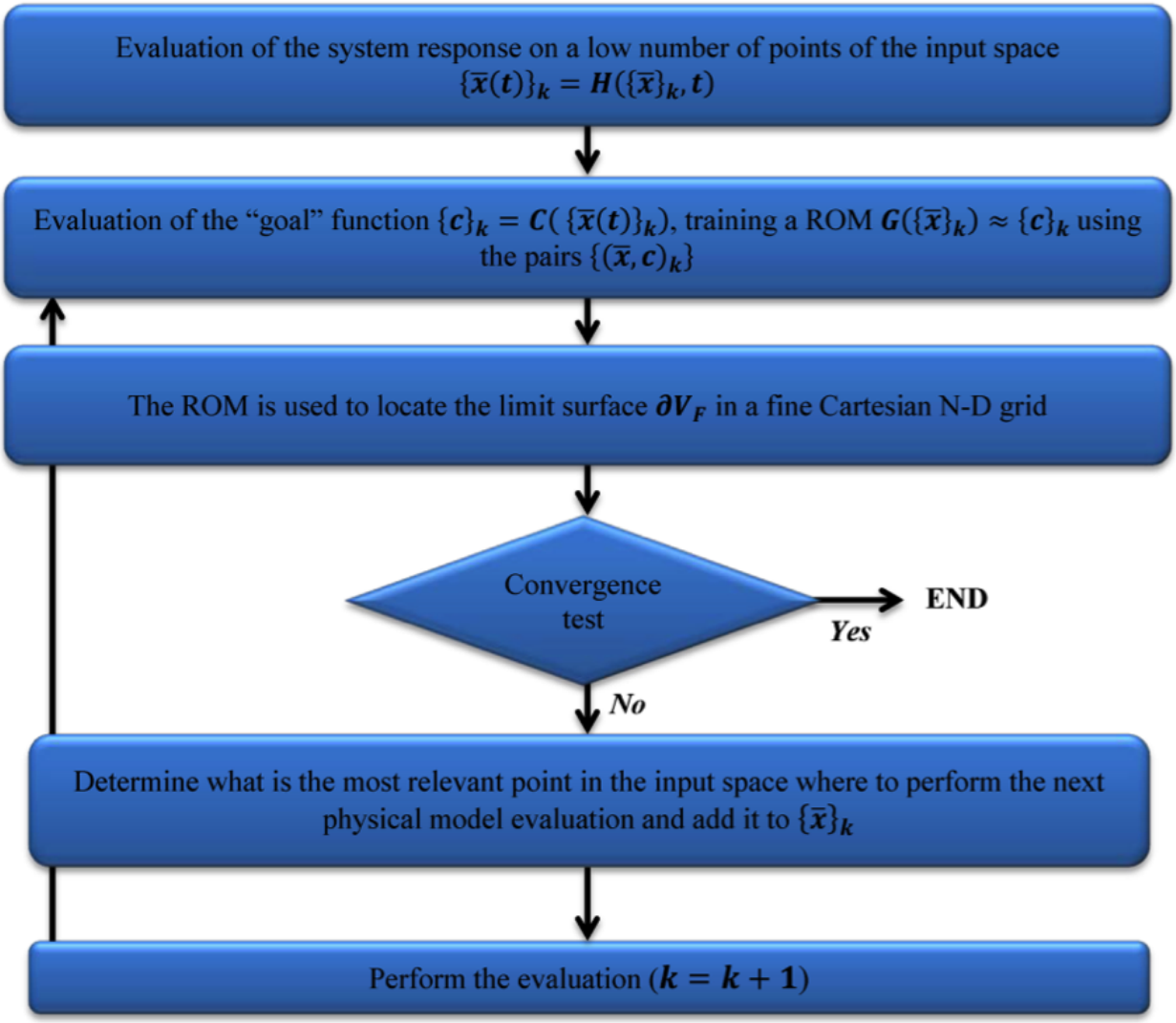

Figure 7. Limit surface search algorithm conceptual scheme.

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, it has been decided to employ 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, the algorithm searches 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, the method will choose the next point employing a Monte Carlo strategy.

4.1.1.2 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 process determines how the evaluation grid is discretized. Before addressing the acceleration scheme, it is important to introduce some concepts on the employed numerical process.
Assume that each of D dimensions of the uncertain domain is discretized with the same number of equally-spaced nodes N (see Figure 8), with discretization size indicated by h_i. Hence, the Cartesian grid contains N^D individual nodes, indexed through the multi-index vector $\overline{j} = (j_{i=1\to D}), j_i \leq N \forall i$. Introducing the vectors $\overline{I} = (1, ..., 1)$ and $\overline{N} = (N, ..., N)$, the "goal" function is expressed on this N- D grid as:

$$C(\overline{x}) = \sum_{\overline{j}=\overline{I}}^{\overline{N}} \varphi_{\overline{j}}(\overline{x}) C\left(\overline{x}_{\overline{j}}\right) \tag{96}$$

where $\varphi_{\overline{j}}$ is the characteristic function of the hyper-volume $\Omega_{\overline{j}}$ surrounding the node $\overline{x}_{\overline{j}}$:

$$\varphi_{\overline{j}}(\overline{x}) = \begin{cases} 1 & if\ \overline{x} \in \Omega_{\overline{j}} \\ 0 & if\ \overline{x} \notin \Omega_{\overline{j}} \end{cases} \tag{97}$$

where:

$$\Omega_{\overline{j}} = \prod_{i=1}^{D} \left[x_{j_i} - \frac{h_i}{2}, x_{j_i} + \frac{h_i}{2} \right] \tag{98}$$

The probability of the uncertain parameters is expressed as:

$$pdf_{\overline{X}}(\overline{x}) = \sum_{\overline{j}=\overline{I}}^{\overline{N}} \varphi_{\overline{j}}(\overline{x}) pdf_{\overline{X}}\left(\overline{x}_{\overline{j}}\right) \tag{99}$$

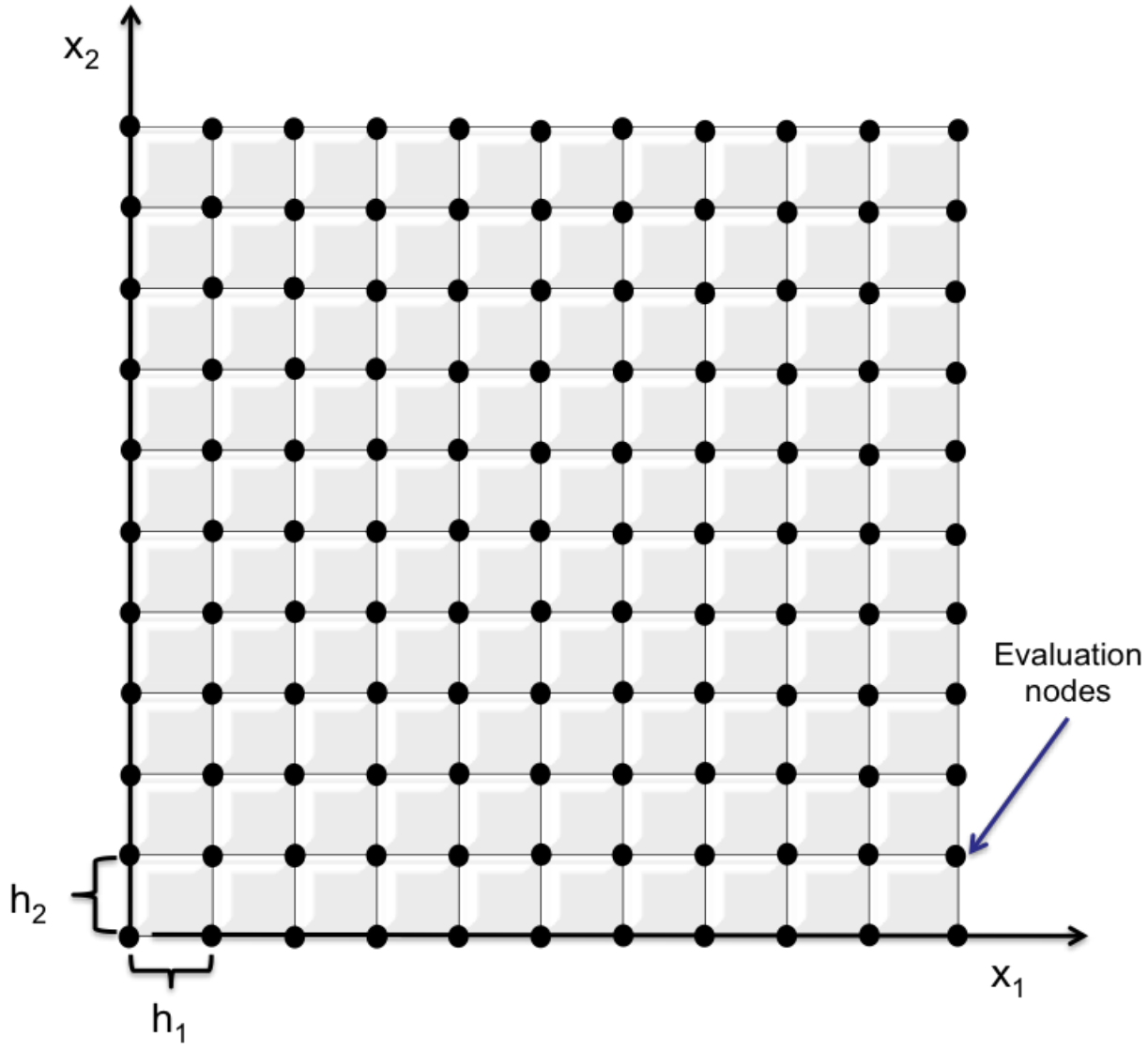

Figure 8. Discretization grid.

Following the approach briefly explained in Section 4.1.1, the probability of the event (e.g., failure) could be expressed as:

$$P_F = \left(\prod_{i=1}^{D} h_i \right) \sum_{\overline{j}=\overline{I}}^{\overline{N}} pdf_{\overline{X}} \left(\overline{x}_{\overline{j}} \right) C \left(\overline{x}_{\overline{j}} \right) \tag{100}$$

Under certain assumptions, the concept of active hyper-volume V_A as the region of the input space identified by the support of the uncertain parameters' probability density functions $pdf_{\overline{X}}(\overline{x})$ could

be introduced; Equation 100 is recast, using a Taylor expansion, as follows:

$$P_F = \int_V C\left(\overline{x}\right) pdf_{\overline{X}}\left(\overline{x}\right) d\overline{x} = \int_{V_A} C\left(\overline{x}\right) \left[\sum_{\overline{j}=\overline{I}}^{\overline{N}} \varphi_{\overline{j}}\left(\overline{x}\right) \left(pdf_{\overline{X}}\left(\overline{x}_{\overline{j}}\right) + \sum_{i=1}^{D} \frac{\partial pdf_{\overline{X}}}{\partial x_i}|_{\overline{x}_{\overline{j}}} \left(x_i - x_{j_i}\right) \right) \right] d\overline{x} \tag{101}$$

And, considering the evaluation grid as:

$$P_F = \sum_{\substack{\overline{j}=\overline{I} \\ \overline{x}_{\overline{j}} \in V_A}}^{\overline{N}} \int_{\overline{x}_{\overline{j}}-\overline{h}/2}^{\overline{x}_{\overline{j}}+\overline{h}/2} C\left(\overline{x}\right) \left[\sum_{\overline{j}=\overline{I}}^{\overline{N}} \varphi_{\overline{j}}\left(\overline{x}\right) \left(pdf_{\overline{X}}\left(\overline{x}_{\overline{j}}\right) + \sum_{i=1}^{D} \frac{\partial pdf_{\overline{X}}}{\partial x_i}|_{\overline{x}_{\overline{j}}} \left(x_i - x_{j_i}\right) \right) \right] d\overline{x} \tag{102}$$

At this point, it is possible to label, in the active hyper-volume, the sub- domain identified by the nodes where the "goal" function $C(\overline{x})$ changes its value (the frontier nodes between the region where $C(\overline{x}) = 1$ and $C(\overline{x}) = 0$) $V_A \cap V_{\partial V_F}$.
Consequentially, it is possible to identify the sub-domains in which the "goal" function $C(\overline{x})$ is equal to 0 ($V_A \cap V_{\partial V_{C(\overline{x})=0}} \notin V_A \cap V_{\partial V_F}$):

$$\sum_{\substack{\overline{j}=\overline{I} \\ \overline{x}_{\overline{j}} \in V_A \cap V_{C(\overline{x})=0}}}^{\overline{N}} \int_{\overline{x}_{\overline{j}}-\overline{h}/2}^{\overline{x}_{\overline{j}}+\overline{h}/2} C\left(\overline{x}\right) \left(pdf_{\overline{X}}\left(\overline{x}_{\overline{j}}\right) + \sum_{i=1}^{D} \frac{\partial pdf_{\overline{X}}}{\partial x_i}|_{\overline{x}_{\overline{j}}} \left(x_i - x_{j_i}\right) \right) d\overline{x} \tag{103}$$

in which the "goal" function $C(\overline{x})$ is equal to 1 ($V_A \cap V_{\partial V_{C(\overline{x})=1}} \notin V_A \cap V_{\partial V_F}$):

$$\begin{aligned} &\sum_{\substack{\overline{j}=\overline{I} \\ \overline{x}_{\overline{j}} \in V_A \cap V_{C(\overline{x})=1}}}^{\overline{N}} \int_{\overline{x}_{\overline{j}}-\overline{h}/2}^{\overline{x}_{\overline{j}}+\overline{h}/2} C\left(\overline{x}\right) \left(pdf_{\overline{X}}\left(\overline{x}_{\overline{j}}\right) + \sum_{i=1}^{D} \frac{\partial pdf_{\overline{X}}}{\partial x_i}|_{\overline{x}_{\overline{j}}} \left(x_i - x_{j_i}\right) \right) d\overline{x} = \\ &= \sum_{\substack{\overline{j}=\overline{I} \\ \overline{x}_{\overline{j}} \in V_A \cap V_{C(\overline{x})=1}}}^{\overline{N}} \int_{\overline{x}_{\overline{j}}-\overline{h}/2}^{\overline{x}_{\overline{j}}+\overline{h}/2} \left(pdf_{\overline{X}}\left(\overline{x}_{\overline{j}}\right) + \sum_{i=1}^{D} \frac{\partial pdf_{\overline{X}}}{\partial x_i}|_{\overline{x}_{\overline{j}}} \left(x_i - x_{j_i}\right) \right) d\overline{x} \end{aligned} \tag{104}$$

Equation 102 is now expressed as:

$$\begin{aligned} P_F = & \sum_{\substack{\overline{j}=\overline{I} \\ \overline{x}_{\overline{j}} \in V_A \cap V_{C(\overline{x})=1}}}^{\overline{N}} \left(\prod_{i=1}^{D} h_i\right) \; pdf_{\overline{X}}(\overline{x}_{\overline{j}}) + O(h^{N+1}) + \\ + & \sum_{\substack{\overline{j}=\overline{I} \\ \overline{x}_{\overline{j}} \in V_A \cap V_{\partial V_f}}}^{\overline{N}} \int_{\overline{x}_{\overline{j}}-\overline{h}/2}^{\overline{x}_{\overline{j}}+\overline{h}/2} C\left(\overline{x}\right) \left(pdf_{\overline{X}}\left(\overline{x}_{\overline{j}}\right) + \sum_{i=1}^{D} \frac{\partial pdf_{\overline{X}}}{\partial x_i}|_{\overline{x}_{\overline{j}}} \; (x_i - x_{j_i}) \right) d\overline{x} \end{aligned} \tag{105}$$

As inferred from Equation 105, the process is bounded if the surface area-to-volume ratio (amount of surface area per unit volume) is in favor of the volume:

$$\sum_{\substack{\overline{j}=\overline{I} \\ \overline{x}_{\overline{j}} \in V_A \cap V_{C(\overline{x})=1}}}^{\overline{N}} \left(\prod_{i=1}^{D} h_i\right) \; pdf_{\overline{X}}(\overline{x}_{\overline{j}}) \gg \sum_{\substack{\overline{j}=\overline{I} \\ \overline{x}_{\overline{j}} \in V_A \cap V_{\partial V_f}}}^{\overline{N}} \left| \int_{\overline{x}_{\overline{j}}-\overline{h}/2}^{\overline{x}_{\overline{j}}+\overline{h}/2} pdf_{\overline{X}}\left(\overline{x}_{\overline{j}}\right) \right| d\overline{x} \tag{106}$$

If the grid is built in the transformed space of probability (i.e., replacing the measure $d\overline{x}$ with $d\overline{\mu} \; pdf_{\overline{X}}\left(\overline{x}_{\overline{j}}\right)$ the condition expressed in Equation 106 is reduced:

$$number \;\; nodes \in V_A \cap V_{C(\overline{x})=1} \gg number \;\; nodes \in V_A \cap V_{\partial V_F} \tag{107}$$

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).

To do that, the step size in probability should be significantly reduced ($h_i^p \rightarrow 0^+$). 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 recast the iterative process in two different sub-sequential 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 ($V_A \cap V_{\partial V_F}$) and, consequentially, converging on the new refined grid.
The iteration scheme is graphically shown in Figure 9. In more detail, the iterative scheme could be summarized through the following steps:

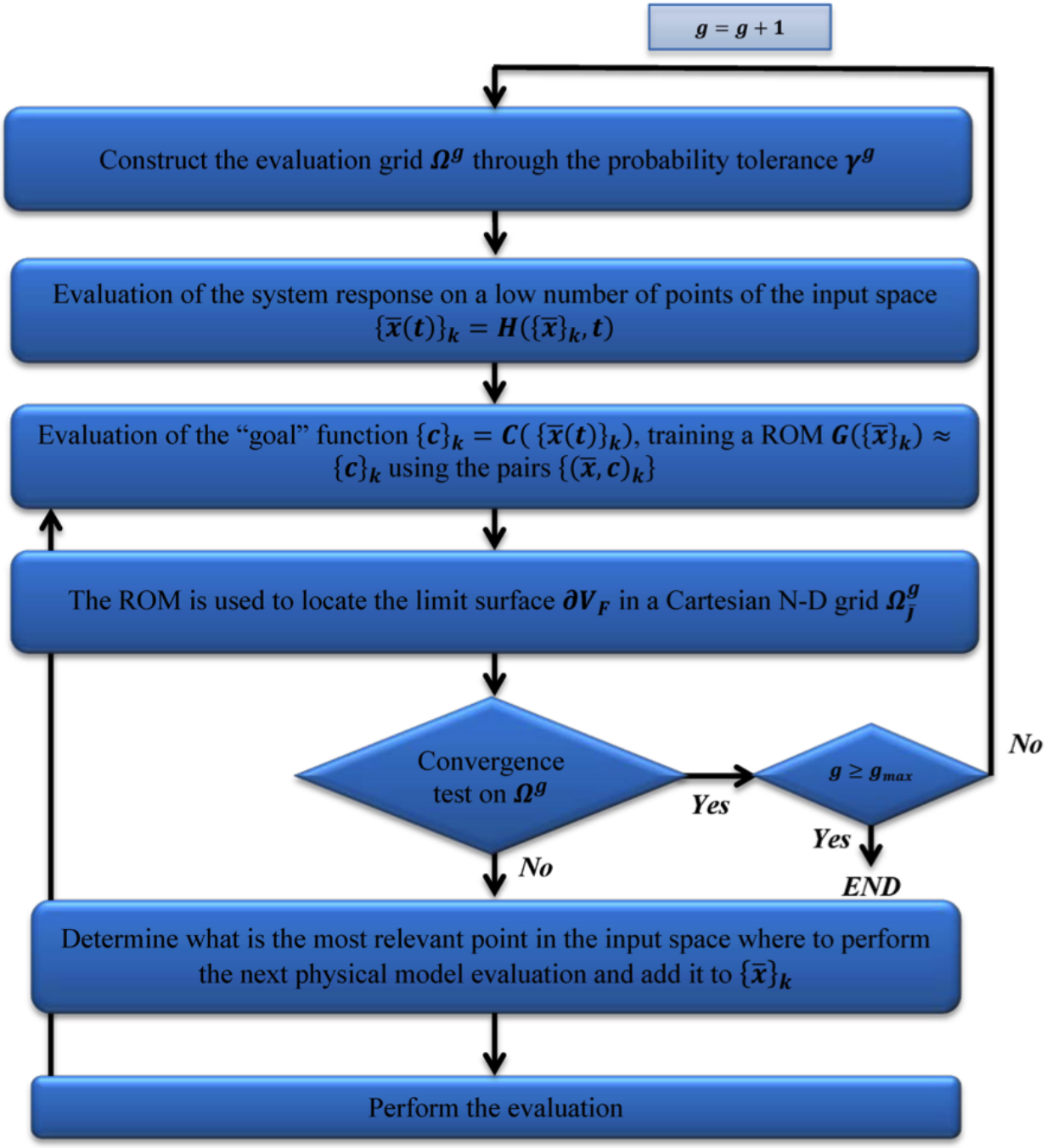

Figure 9. Multi-grid limit surface search scheme.

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 98, 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 $\{\overline{x}_k\}$ are selected via one of the forward sampling strategies (e.g., stratified or Monte Carlo).

4. The high fidelity code is used to compute the status of the system for the set of points in the input set: $\left\{\overline{\theta}(t)\right\}_k = H\left(\{\overline{x}\}_k, t\right)$.

5. The "goal" function is evaluated at the phase space coordinate of the system: $\{c\}_k = C\left(\left\{\overline{\theta}(t)\right\}_k\right)$.

6. The set of pairs $\{(\overline{x}, c)_k\}$ are used to train a ROM of type classifier, $G\left(\{\overline{x}_k\}\right)$.

7. The ROM classifier is used to predict the values of the "goal" function for all the $N^{g=1}$ nodes of the N-D grid in the domain space:

$$\left(G\left(\{\overline{x}\}_j\right) \sim \{c\}_j, j = 1, ..., N^{g=1}\right) \tag{108}$$

8. The values of the "goal" function are used to determine the LS location based on the change of values of $\{c\}_j$:

$$\{c\}_j \rightarrow \partial V_F \tag{109}$$

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 Ω^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 $\Omega^{g=1}$, all the cells that lie on the frontier of the LS ($V_A \cap V_{\partial V_F}$) are refined to contain an amount of probability equal to $\gamma^{g=2}$.

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 9, the algorithm consists in searching the location of the LS proceeding with subsequential refinement of the sub-domain, 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 sub- sequential ROM training process can be regularized, since the LS between an iteration and the other can move, at maximum, within the refinement domain.

5 Reduced Order Modeling

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 $H(\overline{x})$ (see Section 2), which, from a practical point of view, represents the outcome of the system based on the initial conditions $\overline{x}$. Now, sample the domain of variability of the initial conditions $\overline{x}$ to create a set of N realizations of the input and response space $((\overline{x}_i, H(\overline{x}_i)), i = 1, N)$, named "training" set. Based on the data set generated, it is possible to construct a mathematical representation $G(\overline{x} : \overline{x}_i)$ of the real system $H(\overline{x})$, which will approximate its response (see Figure 10):

$$G(\overline{x}) : \overline{x}_i \rightarrow G(\overline{x}_i) \cong H(\overline{x}_i) \tag{110}$$

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 here after 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 110, a classifier can be formally represented as follows (see Figure 11):

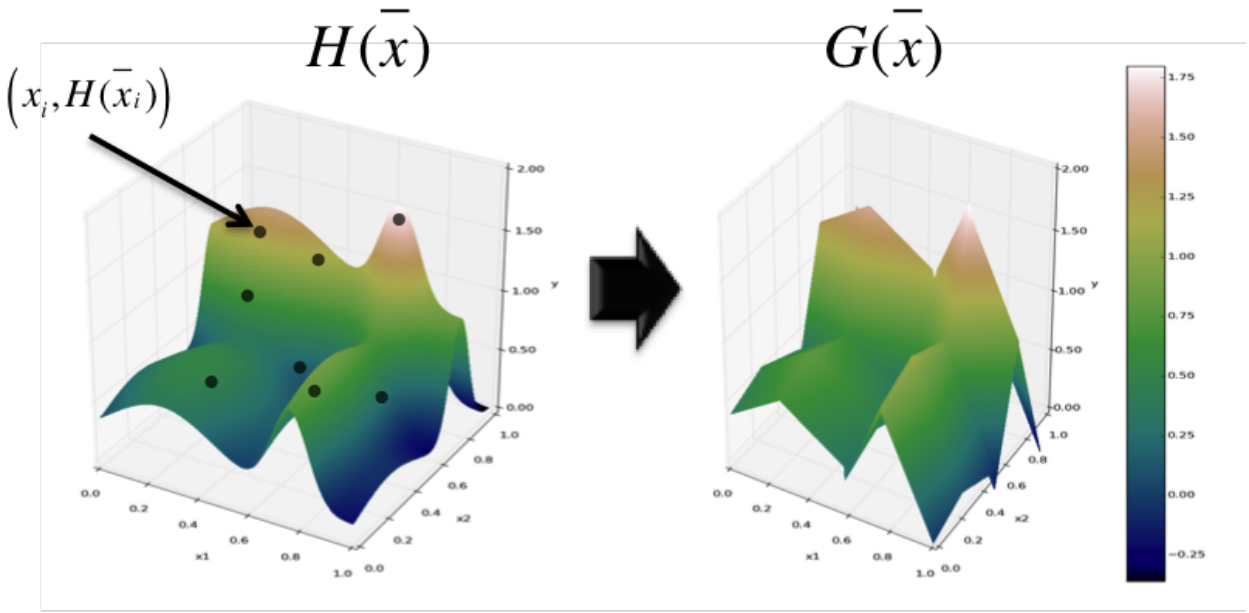

Figure 10. Example of reduced order model representation of physical system (regression).

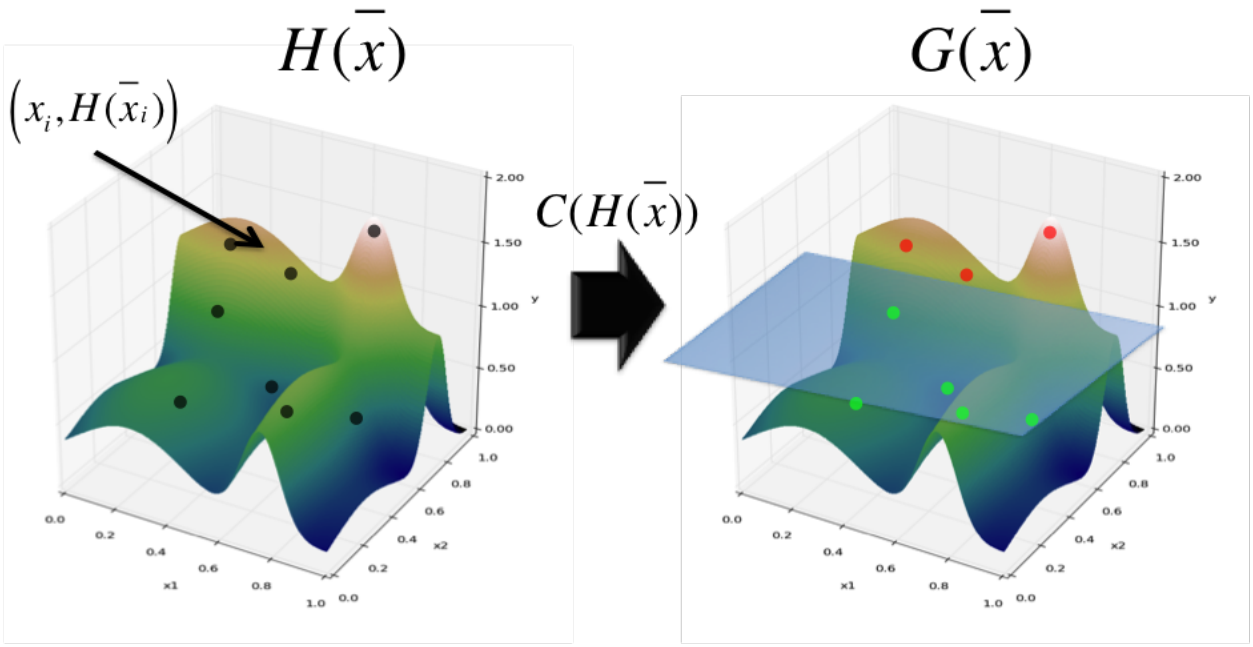

Figure 11. Example of reduced order model representation of physical system (classifier).

$$G\left(\overline{x}\right) : \overline{x}_i \rightarrow G\left(\overline{x}_i\right) \cong C\left(H\left(\overline{x}_i\right)\right) \tag{111}$$

The function $C\left(H\left(\overline{x}_i\right) = \overline{\theta}\right)$ is the so called "goal" function that is able to recast the response of the system $H\left(\overline{x}_i\right)$ into a binary form (e.g., failure/success). As an example, referring to Figure 11, the "goal" function would be:

$$C\left(\overline{\theta}\right) = \begin{cases} 1 & if\ \overline{\theta} > 1.0 \\ 0 & if\ \overline{\theta} \leq 1.0 \end{cases} \tag{112}$$

Hence, the ROM of type classifier $G\left(\overline{x}\right)$ will operate in the space transformed through the "goal" function $C\left(\overline{\theta}\right)$.
The classifiers and regressors can be categorized into two main classes:

- Model-based algorithms
- 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.

Examples of such ROM type are Support Vector Machines (SVMs), Kriging-based regressors, 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. Examples of such ROM type are nearest neighbors and decision trees.

It is important to NOTE that RAVEN uses a Z-score normalization of the training data before constructing most of the ROMs:

$$\mathbf{X} = \frac{(\mathbf{X} - \mu)}{\sigma} \tag{113}$$

In order to identify which ROMs get trained with data normalized by the previous reported normalization approach, please refer to the RAVEN user manual [15].

RAVEN has support of several different ROMs, such as:

1. *Nearest Neighbors approaches*
2. *Support Vector Machines*
3. *Inverse Weight regressors*
4. *Spline regressors* , etc.

In this section only few of them are going to be explained.

5.1 Gaussian Process Models

Gaussian Processes (GPs) [16] are algorithms that extend multivariate Gaussian distributions to infinite dimensionality. A Gaussian process generates a data set located throughout some domain such that any finite subset of the range follows a multivariate Gaussian distribution. Now, the n observations in an arbitrary data set, $y = y_1, \ldots, y_n$, can always be imagined as a single point sampled from some multivariate (n-variate) Gaussian distribution. What relates one observation to another in such cases is just the covariance function, $k(x, x')$. A popular choice is the squared exponential:

$$k(x, x') = \sigma_f^2 exp\left[\frac{-(x - x')^2}{2l^2}\right] \tag{114}$$

where the maximum allowable covariance is defined as σ_f^2; this should be high for functions that cover a broad range on the y axis. If $x \simeq x'$, then $k(x, x')$ approach this maximum meaning $f(x)$

is very correlated to $f(x')$. On the other hand, if x is very distant from x', then $k(x, x') \simeq 0$ (i.e., the two points cannot see each other. So, for example, during interpolation at new x values, distant observations will have negligible effect). How much effect this separation has will depend on the length parameter l. Each observation y can be thought of as related to an underlying function $f(x)$ through a Gaussian noise model:

$$y = f(x) + N(0, \sigma_n^2) \tag{115}$$

The new kernel function can be written as:

$$k(x, x') = \sigma_f^2 exp\left[\frac{-(x - x')^2}{2l^2}\right] + \sigma_n^2 \delta(x, x') \tag{116}$$

So given n observations y, the objective is to predict the value y_* at the new point x_*. This process is performed by following this sequence of steps:

1. Calculate three matrices:

$$K = \begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \dots & k(x_n, x_n) \end{bmatrix} \tag{117}$$

$$K_* = \begin{bmatrix} k(x_*, x_1) & \dots & k(x_*, x_n) \end{bmatrix} \tag{118}$$

$$K_{**} = k(x_*, x_*) \tag{119}$$

2. The basic assumption of GPM is that:

$$\begin{bmatrix} y \\ y_* \end{bmatrix} = \mathcal{N}(0, \begin{bmatrix} K & K_*^T \\ K_* & K_{**} \end{bmatrix}) \tag{120}$$

3. The estimate $\bar{y_*}$ for y_* is the mean of this distribution

$$\bar{y_*} = K_* K^{-1} y \tag{121}$$

4. The uncertainty associated to the estimate $\bar{y_*}$ can be expressed in terms of variance of y_*:

$$var(y_*) = K_{**} - k_* K^{-1} K_*^T \tag{122}$$

5.2 Support Vector Machines

The Support Vector Machine (SVM) [17] classifier is a methodology that aims to determine the optimal separation hyperplane between data sets having different labels. The training data consist

of N data points (x_i, y_i) $i = 1, \ldots, N$ where $x_i \in \mathbb{R}^M$ and $y_i \in -1, 1$. Assuming a linear property of the hyperplane then its definition is:

$$\left\{x : f(x) = x^T\beta + \beta_0 = 0\right\} \tag{123}$$

where β is a unit vector.

The SVM parameters β and β_0 are determined by solving this optimization problem:

$$\begin{cases} \underset{\beta,\beta_0}{min} \|\beta\| \\ \text{subject to } y_i(x_i^T\beta + \beta_0) \geq 1, \quad i = 1, \ldots, N \end{cases} \tag{124}$$

Once the SVM parameters β and β_0 are determined then the classification of a new point $\bar{x}$ is given by:

$$G(\bar{x}) = sign(\bar{x}^T\beta + \beta_0) \tag{125}$$

5.3 KNN Classifier and KNR Regressor

The K Nearest Neighbor algorithm [18] (KNN) is a non-parametric method used for both regression and classification. The only input parameter is the variable K which indicates the number of neighbors to be considered in the classification/regression process. The special case where the class is predicted to be the class of the closest training sample (i.e. when $K = 1$) is called the nearest neighbor algorithm. In binary (two class) classification problems, it is helpful to choose k to be an odd number as this avoids tied votes. The output depends on whether KNN is used for classification or regression:

- In KNN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its K nearest neighbors (K is a positive integer, typically small). If $K = 1$, then the object is simply assigned to the class of that single nearest neighbor.
- In KNN regression, the output is the property value for the object. This value is the average of the values of its K nearest neighbors.

Both for classification and regression, it can be useful to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of $1/d$, where d is the distance to the neighbor.

5.4 Multi-Dimensional Interpolation

This section covers the methods that have been implemented in the CROW statistical library:

- Shepard's Method (see Section 5.4.1)
- Multi-Dimensional Spline method (see Section 5.4.2).

These two methods are interpolation methods that can be used in any dimension. In RAVEN they are employed in two major applications:

1. ROMs
2. Multi-dimensional distributions.

For both applications, given a set of N data points (x_i, u_i) $i = 1, \ldots, N$ where x_i are the coordinate in the input space $D \subset \mathbb{R}^M$ and $u_i \in \mathbb{R}$ is the outcome, the methods predicts the outcome $\tilde{u}$ for a new coordinate $\tilde{x} \in \mathbb{R}^n$.

5.4.1 Shepard's Method

The Shepard interpolator [19] is also know as Inverse Distance Weighting (IDW) interpolator. The starting point is a set of N data points (x_i, u_i) for $i = 1, \ldots, N$. The Inverse-Weight interpolator can be represented as a function $f_{IDW}(x)$ that, given a new coordinate in the input space x, generates a prediction on u such that

$$u : x \in \mathbb{R}^M \rightarrow f_{IDW}(x) \in \mathbb{R} \tag{126}$$

based on the distance $d(x, x_i)$ in the euclidean space between x and x_i.

Such prediction $u = f_{IDW}(x)$ is performed by summing all data points x_i $i = 1, \ldots, N$ weighted by a weighting parameter $w_i(x)$ as follows:

$$f_{IDW}(x) = \begin{cases} \sum_{i=1}^{N} w(x_i) u_i & \text{if } d(x, x_i) \neq 0 \\ u_i & \text{if } d(x, x_i) = 0 \end{cases} \tag{127}$$

where

$$w(x_i) = \frac{w_i}{\sum_{i=1}^{N} w_i} \tag{128}$$

and

$$w_i = \left(\frac{1}{d(x, x_i)} \right)^p \tag{129}$$

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

5.4.2 Multi-Dimensional Spline

The Multi-Dimensional Spline (MDS) [20] is a method that requires the sampled points x_i to be lying in multi-dimensional cartesian grid. A generic grid Δ_m for each dimension m will be indicated as follows:

$$\Delta_m = \{x_{0_m}, x_{1_m}, \ldots, x_{p_m}\} \text{ for } m = 1, \ldots, M \tag{130}$$

This methods construct a M-dimensional cubic spline so that, given a coordinate in the input space $x = (x_1, x_2, \ldots, x_M)$, generates a prediction on u such that

$$u : x \in \mathbb{R}^M \rightarrow f_{MDS}(x) \in \mathbb{R} \tag{131}$$

where

$$f_{MDS}(x) = \sum_{i_1=1}^{p_1+3} \sum_{i_2=1}^{p_2+3} \cdots \sum_{i_M=1}^{p_M+3} c_{i_1,i_2,\ldots,i_p} \prod_{m=1}^{M} u_{i_j}(x_m) \tag{132}$$

where

$$u_{i_j}(x_m) = \Phi\left(\frac{x_m - x_{0_m}}{h_j} + 2 - i_j\right) \tag{133}$$

The cubic kernel $\Phi(t)$ is defined as:

$$\Phi(t) = \begin{cases} (2 - |t|)^3 & 1 \leq |t| \leq 2 \\ 4 - 6\,|t|^2 + 3\,|t|^3 & |t| \leq 1 \\ 0 & \text{elsewhere} \end{cases} \tag{134}$$

The set of $\prod_{m=1}^{M}(p_m+3)$ coefficients $c_{i_1,i_2,\ldots,i_p}$ is determined when the interpolator is initialized.

6 Statistical Analysis

One of the most assessed ways to investigate the impact of the intrinsic variation of the input space is through the computation of statistical moments and linear correlation among variables/parameters/FOMs.

As shown in Section 3, RAVEN employs several different sampling methodologies to explore the response of a model subject to uncertainties. In order to correctly compute the statistical moments a weight-based approach is used. Each *Sampler* in RAVEN associate to each "sample" (i.e. realization in the input/uncertain space) a **weight** to represent the *importance* of the particular combination of input values from a statistical point of view (e.g., reliability weights). These weights are used in subsequential steps in order to compute the previously listed statistical moments and correlation metrics.
In the following subsections, the formulation of these statistical moments is reported.

6.1 Expected Value

The expected value represents one of the most fundamental metrics in probability theory: it represents a measurement of the center of the distribution (mean) of the random variable. From a practical point of view, the expected value of a discrete random variable is the probability-weighted average of all possible values of the subjected variable. Formally, the expected value of a random variable X:

$$\begin{aligned} \mathbb{E}(X) = \mu = \textstyle\sum_{x\in\chi} x pdf_X(x) \quad & \text{if } X \text{ discrete} \\ \mathbb{E}(X) = \mu = \textstyle\int_{x\in\chi} x pdf_X(x) \quad & \text{if } X \text{ continuous} \end{aligned} \tag{135}$$

In RAVEN, the expected value (i.e. first central moment) is computed as follows:

$$\begin{aligned} \mathbb{E}(X) = \mu \approx \overline{x} = \tfrac{1}{n}\textstyle\sum_{i=1}^{n} x_i \quad & \text{if random sampling} \\ \mathbb{E}(X) = \mu \approx \overline{x} = \tfrac{1}{V_1}\textstyle\sum_{i=1}^{n} w_i x_i \quad & \text{otherwise} \end{aligned} \tag{136}$$

where:

- w_i is the weight associated with the sample i
- n are the total number of samples
- $V_1 = \sum_{i=1}^{n} w_i$.

6.2 Standard Deviation and Variance

The variance (σ^2) and standard deviation (σ) of X are both measures of the spread of the distribution of the random variable about the mean. Simplistically, the variance measures how far a set of realizations of a random variable are spread out. The standard deviation is the square root of the variance. The standard deviation has the same unit of the original data, and hence is comparable to deviations from the mean.
Formally:

$$\begin{aligned} \sigma^2(X) &= \mathbb{E}\left([X - \mathbb{E}(X)]^2\right) = \int_{x\in\chi}(x-\mu)^2 pdf(x)dx \quad \text{if } X \text{ continuous} \\ \sigma^2(X) &= \mathbb{E}\left([X - \mathbb{E}(X)]^2\right) = \sum_{x\in\chi}(x-\mu)^2 pdf(x) \qquad \text{if } X \text{ discrete} \\ \sigma(X) &= \mathbb{E}\left([X - \mathbb{E}(X)]\right) = \sqrt{\sigma^2(X)} \end{aligned} \tag{137}$$

In RAVEN, variance (i.e., second central moment) and standard deviation are computed as follows:

$$\begin{aligned} \mathbb{E}\left([X - \mathbb{E}(X)]^2\right) &\approx m_2 = \tfrac{1}{n}\textstyle\sum_{i=1}^{n}(x_i - \overline{x})^2 \qquad \text{if random sampling} \\ \mathbb{E}\left([X - \mathbb{E}(X)]^2\right) &\approx m_2 = \tfrac{1}{V_1}\textstyle\sum_{i=1}^{n} w_i(x_i - \overline{x})^2 \qquad \text{otherwise} \\ \mathbb{E}\left([X - \mathbb{E}(X)]^2\right) &\approx s = \sqrt{m_2} \end{aligned} \tag{138}$$

where:

- w_i is the weight associated with the sample i
- n are the total number of samples
- $V_1 = \sum_{i=1}^{n} w_i$.

RAVEN performs an additional correction of variance to obtain an unbiased estimation with respect to the sample-size [21]:

$$\begin{aligned} \mathbb{E}\left([X - \mathbb{E}(X)]^2\right) &\approx M_2 = \frac{n}{n-1}m_2 \qquad \text{if random sampling} \\ \mathbb{E}\left([X - \mathbb{E}(X)]^2\right) &\approx M_2 = \frac{V_1^2}{V_1^2 - V_2}m_2 \quad textotherwise \end{aligned} \tag{139}$$

$$S = \sqrt{M_2} \tag{140}$$

where:

- w_i is the weight associated with the sample i

- n are the total number of samples
- $V_1 = \sum_{i=1}^{n} w_i^1$.
- $V_2 = \sum_{i=1}^{n} w_i^2$.

It is important to notice that S is not an unbiased estimator.

6.3 Skewness

The Skewness is a measure of the asymmetry of the distribution of a real-valued random variable about its mean. Negative skewness indicates that the tail on the left side of the distribution is longer or fatter than the right side. Positive skewness indicates that the tail on the right side is longer or fatter than the left side. From a practical point of view, the skewness is useful to identify distortion of the random variable with respect to the Normal distribution function.
Formally,

$$\gamma_1 = \mathbb{E}\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] = \frac{\mathbb{E}\left[(X-\mu)^3\right]}{\left(\mathbb{E}\left[(X-\mu)^2\right]\right)^{3/2}} \tag{141}$$

In RAVEN, the skewness is computed as follows:

$$\begin{aligned} \mathbb{E}\left[\left(\tfrac{X-\mu}{\sigma}\right)^3\right] &\approx \frac{m_3}{m_2^{3/2}} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^2\right)^{3/2}} \qquad \text{if random sampling} \\ \mathbb{E}\left[\left(\tfrac{X-\mu}{\sigma}\right)^3\right] &\approx \frac{m_3}{m_2^{3/2}} = \frac{\frac{1}{V_1}\sum_{i=1}^{n} w_i\times(x_i-\overline{x})^3}{\left(\frac{1}{V_1}\sum_{i=1}^{n} w_i\times(x_i-\overline{x})^2\right)^{3/2}} \qquad \text{otherwise} \end{aligned} \tag{142}$$

where:

- w_i is the weight associated with the sample i
- n are the total number of samples
- $V_1 = \sum_{i=1}^{n} w_i$.

RAVEN performs an additional correction of skewness to obtain an unbiased estimation with respect to the sample-size [21]:

$$\begin{aligned} \mathbb{E}\left[\left(\tfrac{X-\mu}{\sigma}\right)^3\right] &\approx \frac{M_3}{M_2^{3/2}} = \frac{n^2}{(n-1)(n-2)} m_3 \times \frac{1}{\left(\frac{n}{n-1}m_2\right)^{3/2}} \qquad \text{if random sampling} \\ \mathbb{E}\left[\left(\tfrac{X-\mu}{\sigma}\right)^3\right] &\approx \frac{M_3}{M_2^{3/2}} = \frac{V_1^3}{V_1^3 - 3V_1V_2 + 2V_3} m_3 \times \frac{1}{\left(\frac{V_1^2}{V_1^2 - V_2}m_2\right)^{3/2}} \qquad \text{otherwise} \end{aligned} \tag{143}$$

where:

- w_i is the weight associated with the sample i
- n are the total number of samples
- $V_1 = \sum_{i=1}^{n} w_i^1$
- $V_2 = \sum_{i=1}^{n} w_i^2$
- $V_3 = \sum_{i=1}^{n} w_i^3$.

6.4 Excess Kurtosis

The Kurtosis [22] is the degree of peakedness of a distribution of a real-valued random variable. In a similar way to the concept of skewness, kurtosis describes the shape of the distribution. The Kurtosis is defined in order to obtain a value of 0 for a Normal distribution. If it is greater than zero, it indicates that the distribution is high peaked; If it is smaller that zero, it testifies that the distribution is flat-topped.
Formally, the Kurtosis can be expressed as follows:

$$\gamma_2 = \frac{\mathbb{E}\left[(X-\mu)^4\right]}{\left(\mathbb{E}\left[(X-\mu)^2\right]\right)^2} \tag{144}$$

In RAVEN, the kurtosis (excess) is computed as follows:

$$\frac{\mathbb{E}\left[(X-\mu)^4\right]}{\left(\mathbb{E}\left[(X-\mu)^2\right]\right)^2} \approx \frac{m_4 - 3m_2^2}{m_2^2} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^4 - 3\left(\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^2\right)^2}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^2\right)^2} \quad \text{if random sampling}$$

$$\frac{\mathbb{E}\left[(X-\mu)^4\right]}{\left(\mathbb{E}\left[(X-\mu)^2\right]\right)^2} \approx \frac{m_4 - 3m_2^2}{m_2^2} = \frac{\frac{1}{V_1}\sum_{i=1}^{n} w_i \times (x_i-\overline{x})^4 - 3\left(\frac{1}{V_1}\sum_{i=1}^{n} w_i \times (x_i-\overline{x})^2\right)^2}{\left(\frac{1}{V_1}\sum_{i=1}^{n} w_i \times (x_i-\overline{x})^2\right)^2} \quad \text{otherwise} \tag{145}$$

where:

- w_i is the weight associated with the sample i
- n are the total number of samples
- $V_1 = \sum_{i=1}^{n} w_i$.

RAVEN performs an additional correction of kurtosis (excess) to obtain an unbiased estimation with respect to the sample-size [21]:

$$\frac{\mathbb{E}\left[(X-\mu)^4\right]}{\left(\mathbb{E}\left[(X-\mu)^2\right]\right)^2} \approx \frac{M_4-3M_2^2}{M_2^2} = \frac{n^2(n+1)}{(n-1)(n-2)(n-3)}m_4 - \frac{3n^2}{(n-2)(n-3)}m_2^2 \quad \text{if random sampling}$$

$$\frac{\mathbb{E}\left[(X-\mu)^4\right]}{\left(\mathbb{E}\left[(X-\mu)^2\right]\right)^2} \approx \frac{M_4-3M_2^2}{M_2^2} = \frac{V_1^2(V_1^4-4V_1V_3+3V_2^2)}{(V_1^2-V_2)(V_1^4-6V_1^2V_2+8V_1V_3+3V_2^2-6V_4)}m_4 - \frac{3V_1^2(V_1^4-2V_1^2V_2+4V_1V_3-3V_2^2)}{(V_1^2-V_2)(V_1^4-6V_1^2V_2+8V_1V_3+3V_2^2-6V_4)}m_2^2 \quad \text{otherwise} \tag{146}$$

where:

- w_i is the weight associated with the sample i
- n are the total number of samples
- $V_1 = \sum_{i=1}^{n} w_i^1$
- $V_2 = \sum_{i=1}^{n} w_i^2$
- $V_3 = \sum_{i=1}^{n} w_i^3$
- $V_4 = \sum_{i=1}^{n} w_i^4$.

6.5 Median

The median of the distribution of a real-valued random variable is the number separating the higher half from the lower half of all the possible values. The median of a finite list of numbers can be found by arranging all the observations from lowest value to highest value and picking the middle value.
Formally, the median m can be cast as the number that satisfy the following relation:

$$P(X \leq m) = P(X \geq m) = \int_{-\infty}^{m} pdf(x)dx = \frac{1}{2} \tag{147}$$

6.6 Percentile

A percentile (or a centile) is a measure indicating the value below which a given percentage of observations in a group of observations fall.

6.7 Covariance and Correlation Matrices

Simplistically, the Covariance is a measure of how much two random variables variate together. In other words, It represents a measurement of the correlation, in terms of variance, among different variables. If the greater values of one variable mainly correspond with the greater values of the other variable, and the same holds for the lesser values (i.e., the variables tend to show similar behavior) the covariance is positive. In the opposite case, when the greater values of one variable mainly correspond to the lesser values of the other (i.e., the variables tend to show opposite behavior) the covariance is negative. Formally, the Covariance can be expressed as

$$\boldsymbol{\Sigma}(\boldsymbol{X}, \boldsymbol{Y}) = \mathbb{E}\left[(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])\,(\boldsymbol{Y} - \mathbb{E}[\boldsymbol{Y}])^T\right] \tag{148}$$

Based on the previous equation, in RAVEN each entry of the Covariance matrix is computed as follows:

$$\begin{aligned} \mathbb{E}[(X - \mathbb{E}[X])\,(Y - \mathbb{E}[Y])] &\approx \tfrac{1}{n}\textstyle\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y) \qquad \text{if random sampling} \\ \mathbb{E}[(X - \mathbb{E}[X])\,(Y - \mathbb{E}[Y])] &\approx \tfrac{1}{V_1}\textstyle\sum_{i=1}^{n} w_i \times (x_i - \mu_x)(y_i - \mu_y) \qquad \text{otherwise} \end{aligned} \tag{149}$$

where:

- w_i is the weight associated with the sample i
- n are the total number of samples
- $V_1 = \sum_{i=1}^{n} w_i$.

The correlation matrix (Pearson product-moment correlation coefficient matrix) can be obtained through the Covariance matrix, as follows:

$$\boldsymbol{\Gamma}(\boldsymbol{X}, \boldsymbol{Y}) = \frac{\boldsymbol{\Sigma}(\boldsymbol{X}, \boldsymbol{Y})}{\sigma_x \sigma_y} \tag{150}$$

As it can be seen, The correlation between X and Y is the covariance of the corresponding standard scores.

6.8 Variance-Dependent Sensitivity Matrix

The variance dependent sensitivity matrix is the matrix of the sensitivity coefficients that show the relationship of the individual uncertainty component to the standard deviation of the reported value for a test item.
Formally:

$$\boldsymbol{\Lambda} = \boldsymbol{\Sigma}(\boldsymbol{X}, \boldsymbol{Y}) vc^{-1}(\boldsymbol{Y}) \tag{151}$$

where:

- $vc^{-1}(\boldsymbol{Y})$ is the inverse of the covariance of the input space.

6.9 Normalized Sensitivity Matrix

The normalized sensitivity matrix is the matrix of the sensitivity coefficients normalized with respect their expected values. This means that this matrix represents, in a linear formulation, the p.u. sensitivity coefficients.
Formally:

$$\boldsymbol{\Lambda} = \left(\boldsymbol{\Sigma}(\boldsymbol{X}, \boldsymbol{Y}) vc^{-1}(\boldsymbol{Y})\right) \times \frac{\mathbb{E}(\boldsymbol{Y})}{\mathbb{E}(\boldsymbol{X})} \tag{152}$$

where:

- $vc^{-1}(\boldsymbol{Y})$ is the inverse of the covariance of the input space.
- $\mathbb{E}(\boldsymbol{Y})$ is the expected value of the input space
- $\mathbb{E}(\boldsymbol{X})$ is the expected value of the output space

7 Data Mining

Data mining is the computational process of discovering patterns in large data sets ("big data") involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use.
RAVEN has support of several different data mining algorithms, such as:

1. *Hierarchical methodologies*
2. *K-Means*
3. *Mean-Shift*, etc.

In this section only few algorithms will be explained

7.1 Clustering

A loose definition of clustering is the process of organizing objects into groups whose members are, in some way, similar. Therefore, a cluster is a collection of objects that are similar to each other and are dissimilar to the objects belonging to other clusters [23, 24].

The similarity criterion is distance. Two or more objects belong to the same cluster if they are "close" according to a specified distance. The approach of using distance metrics to clustering is called distance-based clustering and is used in this work.

The notion of distance implies that the data points lay in a metric space [25]:

Definition 1 (Metric Space) *A metric space is a space X provided with a function* d*:* $f : X \times X \rightarrow \mathbb{R}$ *satisfying the following properties* $\forall \boldsymbol{x}, \boldsymbol{y} \in X$ *:*

- $d(\boldsymbol{x}, \boldsymbol{y}) \geqslant 0$
- $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$
- $d(\boldsymbol{x}, \boldsymbol{y}) \leqslant d(\boldsymbol{x}, \boldsymbol{z}) + d(\boldsymbol{z}, \boldsymbol{y})$

The function $d(\boldsymbol{x}, \boldsymbol{y})$ is usually called the distance function. In a 2-dimensional Euclidean space ($\mathbb{R}^2$), the distance between points can be calculated using the Pythagorean theorem which is the direct application of the Euclidean distance and is a special case of the most general Minkowski

distance $d_2(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$ between two points $\boldsymbol{x} = (x_1, x_2)$ and $\boldsymbol{y} = (y_1, y_2)$ in $\mathbb{R}^2$.

In the literature [25], it is possible to find several types of distances other than the Euclidean and the Minkowski distance as shown in Table 6. The approach of using distance metrics is called distance-based clustering and will be used in this dissertation.

Table 6. Summary of the commonly used measures [25].

Measure	Form
Minkowski distance	$d_n(\boldsymbol{x}, \boldsymbol{y}) = (\sum_{k=1}^{\delta} \lvert x_k - y_k \rvert^n)^{\frac{1}{n}}$
Euclidean distance	$d_2(\boldsymbol{x}, \boldsymbol{y}) = (\sum_{k=1}^{\delta} \lvert x_k - y_k \rvert^2)^{\frac{1}{2}}$
Taxicab distance	$d_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{k=1}^{\delta} \lvert x_k - y_k \rvert$
Supremum distance	$d_0(\boldsymbol{x}, \boldsymbol{y}) = max_k \lvert x_k - y_k \rvert$
Mahalanobis distance	$d_M(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^T S^{-1} (\boldsymbol{x} - \boldsymbol{y})$

From a mathematical viewpoint, the concept of clustering [23] aims to find a partition $\mathbf{C} = \{C_1, \ldots, C_l, \ldots, C_L\}$ of the set of I scenarios $\mathbf{X} = \{\boldsymbol{x_1}, \ldots, \boldsymbol{x_i}, \ldots, \boldsymbol{x_I}\}$ where each scenario $\boldsymbol{x_i}$ is represented as a δ-dimensional vector. Each C_l $(l = 1, \ldots, L)$ is called a cluster. The partition $\mathbf{C}$ of $\mathbf{X}$ is given as follows[1]:

$$\begin{cases} \mathbf{C}_l \neq \varnothing, l = 1, \ldots, L \\ \bigcup_{l=1}^{L} \mathbf{C}_l = \mathbf{X} \end{cases} \tag{153}$$

7.2 Hierarchical Methodologies

These methodologies organize the data set into a hierarchical structure according to a proximity matrix. Each element $d(i, j)$ of this matrix contains the distance between the the i^{th} and the j^{th} cluster center. The final results of this technique is a tree commonly called a dendrogram. This kind

[1] In most clustering algorithms each scenario belongs to only one cluster. However this is not always the case. In fuzzy clustering methodologies [26] a scenario may be allowed to belong to more than one cluster with a degree of membership $u_{i,j} \in [0, 1]$ which represents the member coefficient of the j scenario for the i^{th} cluster and satisfies the following properties:
$\sum_{i=1}^{K} u_{i,j} = 1$, and $\sum_{j=1}^{N} u_{i,j} < N, \forall j$

of representation has the advantages of providing a very informative description and visualization of the data structure even for high values of dimensionality.

The procedure to determine the dendrogram for a data set of I points in an δ-dimensional space is the following:

1. Start the analysis with a set of I clusters (i.e., each point is considered as a cluster).
2. Determine the proximity matrix M (dimension: $I \times I$): $M(i,j) = d(\boldsymbol{x_i}, \boldsymbol{x_j})$ where $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ are the position of the i^{th} and the j^{th} cluster.
3. For each point p find the closest neighbor q from the proximity matrix M
4. Combine the points p and q
5. Repeat Steps 2, 3 and 4 until all the points of the data set are in the same cluster

The advantage of this kind of algorithm is the nice visualization of the results that show the underlying structure of the data set. However, the computational complexity for most of the hierarchical algorithm is of the order of $\mathcal{O}(I^2)$ (where *I* is the number of points in the data set).

7.3 *K*-Means

K-Means clustering algorithms belong to the more general family of Squared Error algorithms. The goal is to partition I data points $\boldsymbol{x_i}$ $(i = 1, \ldots, I)$ into *K* clusters in which each data point maps to the cluster with the nearest mean. The stopping criterion is to find the global minimum of the error squared function χ defined as:

$$\chi = \sum_{i=1}^{K} \sum_{x_j \in C_i} |\boldsymbol{x_j} - \boldsymbol{\mu_i}|^2 \tag{154}$$

where $\boldsymbol{\mu_i}$ is the centroid (i.e., the center) of the cluster C_i.

The procedure to determine the centroids $\boldsymbol{\mu_i}$ of *K* clusters $(C_1, \ldots, C_K)$ is the following:

1. Start with a set of K random centroids distributed in the state space
2. Assign each pattern to the the closest centroid

3. Determine the new K centroids according to the point-centroid membership

$$\mu_i = \frac{1}{N_i} \sum_{\boldsymbol{x_j} \in C_i} \boldsymbol{x_j} \tag{155}$$

where N_i corresponds to the number of of data points in the i^{th} cluster.

4. Repeat Steps 2 and 3 until convergence is met (i.e., until a minima of the χ function is reached)

K-Means algorithm is one of the most popular and used methodologies also due to the fact that is very straightforward to implement and the computational time is directly proportional to the cardinality of data points (i.e., $\mathcal{O}(I)$ where *I* is the number of data points). The main disadvantage is that the algorithm is sensitive to the choice of the initial partition and may converge to a local minimum of the error squared function [27]. Another disadvantage of this algorithm is that is only able to identify clusters having spherical or ellipsoidal geometry. Thus, *K*-Means is not able to identify clusters of points having arbitrary shapes. Moreover, the number of cluster *K* to be obtained is specified by the user prior the clustering process.

7.4 Mean-Shift

The Mean-Shift algorithm [28] is a non-parametric iterative procedure that can be used to assign each point to one cluster center through a set of local averaging operations [28]. The local averaging operations provide empirical cluster centers within the locality and define the vector which denotes the direction of increase for the underlying unknown density function.

The underlying idea is to treat each point $\boldsymbol{x_i}$ $(i = 1, \ldots, I)$ of the dataset as an empirical probability distribution function using kernel $K(\boldsymbol{x}) : \mathbb{R}^{M \cdot K} \rightarrow \mathbb{R}$. This multivariate kernel density resides in a multidimensional space where regions with high data density (i.e., modes) correspond to local maxima of the density estimate $f_I(\boldsymbol{x})$ [29] defined by:

$$f_I(\boldsymbol{x}) = \frac{1}{Ih^d} \sum_{i=1}^{I} K\left(\frac{\boldsymbol{x} - \boldsymbol{x_i}}{h}\right), \tag{156}$$

where $\boldsymbol{x} \in \mathbb{R}^{M \cdot K}$ and h is often referred as the bandwidth associated with the kernel.

The kernel in Equation 156 serves as a weighting function [29] associated with each data point and is expressed as:

$$K(\boldsymbol{x}) = c_k k(\|\boldsymbol{x}\|^2) \tag{157}$$

where $k(x) : [0, \infty] \rightarrow \mathbb{R}$ is referred as the *kernel profile* and c_k is a normalization constant. The profile satisfies the following properties:

- $k(x)$ is non negative
- $k(x)$ is non increasing (i.e., $k(a) \geq k(b)$ if $a < b$)
- $k(x)$ is piecewise continuous and $\int_0^\infty k(x)\,dx < \infty$

In order to estimate the data points with highest probability from an initial estimate (i.e., the modes of $f_I(\boldsymbol{x})$), consider the gradient of the density function $\nabla_x f_I(\boldsymbol{x}) = 0$ [28] where

$$\begin{aligned}\nabla_x f_I(\boldsymbol{x}) &= \frac{2c_k}{Ih^{d+2}} \sum_{i=1}^{I} (\boldsymbol{x} - \boldsymbol{x_i}) k' \left(\| \frac{\boldsymbol{x} - \boldsymbol{x_i}}{h} \|^2 \right) \\ &= \underbrace{\frac{2c_k}{Ih^{d+2}} \left(\sum_{i=1}^{I} g \left(\| \frac{\boldsymbol{x} - \boldsymbol{x_i}}{h} \|^2 \right) \right)}_{A} \underbrace{\left(\frac{\sum_{i=1}^{I} \boldsymbol{x} g \left(\| \frac{\boldsymbol{x} - \boldsymbol{x_i}}{h} \|^2 \right)}{\sum_{i=1}^{I} g \left(\| \frac{\boldsymbol{x} - \boldsymbol{x_i}}{h} \|^2 \right)} - \boldsymbol{x} \right)}_{B}, \end{aligned} \tag{158}$$

which points in the direction of the increase in kernel density estimate. The kernel $K(\boldsymbol{x})$ is also referred to as the shadow of $G(\boldsymbol{x}) = c_g g(\|\boldsymbol{x}\|^2)$ [30] where c_g, similar to c_k, is a normalization constant and $g(x)$ is the derivative of $k(x)$ over x, i.e., $g(x) = k'(x)$. In the equation above, the first term denoted as A is a scalar proportional to the density estimate computed with the kernel $G(\boldsymbol{x})$ and does not provide information regarding where the mode resides. Unlike A, the vector quantity B, which is the second term in the equation above, is difference between the weighted mean

$$m(\boldsymbol{x}) = \frac{\sum_{i=1}^{I} \boldsymbol{x} g(\| \frac{\boldsymbol{x} - \boldsymbol{x_i}}{h} \|^2)}{\sum_{i=1}^{I} g(\| \frac{\boldsymbol{x} - \boldsymbol{x_i}}{h} \|^2)}. \tag{159}$$

and the initial estimate $\boldsymbol{x}$. This term points in the direction of local increase in density using kernel $G(\boldsymbol{x})$, hence provides a means to find the mode of the density. Note that all points used to compute a particular mode are considered to reside in the same cluster.

Since each each data point $\boldsymbol{x_i}$ (or scenario) is considered as an empirical probability distribution function, this consideration allows to include in the scenario clustering analysis also the possible uncertainty associated with each scenario.

7.5 DBSCAN

The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm views clusters as areas of high density of data points. The data points in the low-density areas are seen as noise and border points, which are actually separating the clusters. Clusters found by DBSCAN can be any shape because of this approach. The main element of the DBSCAN algorithm is the

concept of core samples, which are samples that are in areas of high density. Therefore, a cluster is a set of core samples, each close to each other (measured by some distance measure) and a set of non-core samples that are close to a core sample (but are not themselves core samples). There are two parameters to the algorithm: $min_samples$ and eps. Higher $min_samples$ or lower eps indicate higher density necessary to form a cluster. A cluster is a set of core samples, that can be built by recursively by taking a core sample, finding all of its neighbors that are core samples, finding all of their neighbors that are core samples, and so on. A cluster also has a set of non-core samples, which are samples that are neighbors of a core sample in the cluster but are not themselves core samples; these are on the borders of a cluster. The DBSCAN algorithm finds core samples of high density and expands clusters from them. It is good for data, which contains clusters of similar density.

7.6 Dimensionality Reduction

The dimensionality δ of each data point (i.e., each scenario) is equal to the product of the number of variables (i.e., M) chosen to represent each scenario multiplied by the number of times each variable has been sampled. In order to reduce the computational time due to the high data dimensionality, the use of dimensionality reduction techniques was to reduce the number of variables M[2].

The raw data generated by DET methodologies contain the temporal behavior of a vast set of variables (e.g., temperature, pressure). These variables are often heavily correlated and, consequently, the information contained in the set of M variables comprising the full state space can be condensed to a set of N variables where $N < M$. The objective of the dimensionality reduction process is to determine those N variables by finding the correlations among the original M variables[3].

Linear algorithms, such as PCA [31] or multidimensional scaling (MDS) [32], have the advantage that they are easier to implement but they can only identify linear correlation among variables. On the other hand, methodologies such as Local Linear Embedding [33] and ISOMAP [34] are more computationally intensive but they are able to identify non-linear correlations.

Dimensionality reduction is the process of finding a bijective mapping function $\mathfrak{F}$

$$\mathfrak{F} : \mathbb{R}^D \mapsto \mathbb{R}^d \text{ (where } d < D) \tag{160}$$

which maps the data points from the D-dimensional space into a reduced d-dimensional space (i.e. embedding on a manifold) in such a way that the distances between each point and its neighbors are preserved. In our applications $D = M + 1$, i.e. M state variables plus time t.

[2]Other possible options are to reduce the number of sample instants K or to observe the local properties of the covariance matrix S.

[3]Note that those N variables are not necessarily a subset of the original M variables but, more likely, a combination of those M variables.

7.7 Dimensionality Reduction: Linear Algorithms

This section describes the two most important algorithms for dimensionality reduction:

1. PCA (see Section 7.8), and,
2. MDS (see Section 7.9).

7.8 Principal Component Analysis (PCA)

The main idea behind PCA [31] is to perform a linear mapping of the data set onto a lower dimensional space such that the variance of the data in the low-dimensional representation is maximized.

This is accomplished by determining the eigenvectors and their corresponding eigenvalues of the data covariance matrix[4] S. The eigenvectors that correspond to the largest eigenvalues (i.e., the principal components) can be used as a set of basis functions. Thus, the original space is reduced to the space spanned by a few eigenvectors.

The algorithm is very straightforward to implement but, on the other hand, PCA is not able to identify non-linear correlations of more complex data sets.

7.9 Multidimensional Scaling (MDS)

Multidimensional scaling [32] is a popular technique used to analyze the properties of data sets. The scope of this methodology is to find a set of dimensions that preserve distances between data points.

This is performed by:

1. Creating dissimilarity matrix $D = [d_{ij}]$ where d_{ij} is the distance between two points x_i and x_j.
2. Finding the hyper-plane that preserves the dissimilarity matrix D (i.e., the *nearness* of points)

As in PCA analysis, the algorithm can be easily implemented but it is not able to identify non-linear correlations of more complex data sets.

[4]Given a data set in form of a vector Z, rows correspond to data dimensions (D) and columns correspond to data observations (Λ), the covariance matrix S is determined as: $S = \frac{1}{\Lambda - 1} Z'Z$.

Appendices

A Document Version Information

c55a3a27da85f5e393b9f582f9f0fdfe7ba96429 Andrea Alfonsi Fri, 3 May 2019 14:24:36 -0600

References

[1] A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, and B. Kinoshita, "Raven as a tool for dynamic probabilistic risk assessment: Software overview," in *Proceedings of International Conference of mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2013), Sun Valley (Idaho)*, pp. 1247–1261, 2013.

[2] A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, B. 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, Columbia (South Carolina)*, 2013.

[3] C. Rabiti, A. Alfonsi, D. Mandelli, J. Cogliati, and R. Kinoshita, "Deployment and overview of raven capabilities for a probabilistic risk assessment demo for a pwr station blackout," Tech. Rep. INL/EXT-13-29510, Idaho National Laboratory (INL), 2013.

[4] A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, and B. Kinoshita, "Raven and dynamic probabilistic risk assessment: Software overview," in *Proceedings of ESREL European Safety and Reliability Conference (ESREL 2014), Wrocklaw (Poland)*, 2014.

[5] D. Anders, 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," Tech. Rep. INL/EXT-12-25924, Idaho National Laboratory (INL), 2012.

[6] R.-D. development team, "Relap5/mod3.3 code manual," tech. rep., Idaho National Laboratory, October 2015.

[7] J. Devooght and C. Smidts, "Probabilistic reactor dynamics - i: The theory of continuous event trees," *Nuclear Science and Engineering*, vol. 111, pp. 229–240, 1992.

[8] Wiener, "The homogeneous chaos," *American Journal of Mathematics*, vol. 60, pp. 897–936, 1938.

[9] Askey and Wilson, "Some basic hypergeometric orthogonal polynomials that generalize jacobi polynomials," *Memoirs of the American Mathematical Society*, vol. 54, pp. 1–55, 1985.

[10] Xiu and Karniadakis, "The wiener–askey polynomial chaos for stochastic differential equations," *SIAM Journal on Scientific Computing*, vol. 24, no. 2, pp. 619–644, 2002.

[11] Babuska, Nobile, and Tempone, "A stochastic collocation method for elliptic partial differential equations with random input data," *SIAM Journal on Numerical Analysis*, vol. 45, 2007.

[12] Novak and Ritter, "The curse of dimension and a universal method for numerical integration," in *Multivariate approximation and splines* (G. Nürnberger, J. Schmidt, and G. Walz, eds.), vol. 125 of *ISNM International Series of Numerical Mathematics*, pp. 177–187, Birkhäuser Basel, 1997.

[13] Smolyak, "Quadrature and interpolation formulas for tensor products of certain classes of functions," in *Dokl. Akad. Nauk SSSR*, vol. 4, p. 123, 1963.

[14] C. Rabiti, D. M. 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.

[15] C. Rabiti, A. Alfonsi, J. Cogliati, D. Mandelli, R. Kinoshita, and S. Sen, "Raven user manual," tech. rep., Idaho National Laboratory, 2015.

[16] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[17] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, June 1998.

[18] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[19] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proceedings of the 1968 23rd ACM National Conference*, ACM 1968, (New York, NY, USA), pp. 517–524, ACM, 1968.

[20] C. Habermann and F. Kindermann, "Multidimensional spline interpolation: Theory and applications," *Computational Economics*, vol. 30, no. 2, pp. 153–169, 2007.

[21] L. Rimoldini, "Weighted skewness and kurtosis unbiased by sample size," tech. rep., Astronomical Observatory of the University of Geneva, April 2013.

[22] A. M. and S. I. A, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, 9th ed., 1972.

[23] X. Rui and Ii, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, pp. 645–678, May 2005.

[24] D. Mandelli, A. Yilmaz, T. Aldemir, K. Metzroth, and R. Denning, "Scenario clustering and dynamic probabilistic risk assessment," *Reliability Engineering & System Safety*, vol. 115, pp. 146–160, 2013.

[25] B. Mendelson, *Introduction to Topology*. New York (NY), USA: Dover Publications, 1990.

[26] E. Zio and F. D. Maio, "Processing dynamic scenarios from a reliability analysis of a nuclear power plant digital instrumentation and control system," *Annals of Nuclear Energy*, vol. 36, pp. 1386–1399, 2009.

[27] A. K. Jain, K. Dubes, and C. Richard, *Algorithms for clustering data*. Upper Saddle River, NJ (USA): Prentice-Hall, Inc., 1988.

[28] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on Information Theory*, vol. 21, no. 1, pp. 32–40, 1975.

[29] T. Cacoullos, "Estimation of a multivariate density," *Annals of the Institute of Statistical Mathematics*, vol. 18, no. 1, pp. 179–189, 1966.

[30] Y. A. Sheikh, E. Khan, and T. Kanade, "Mode-seeking by medoidshifts," in *Eleventh IEEE International Conference on Computer Vision (ICCV 2007)*, no. 1, October 2007.

[31] I. T. Jolliffe, *Principal Component Analysis*. Springer, second ed., October 2002.

[32] I. Borg and P. Groenen, *Modern Multidimensional Scaling: Theory and Applications*. Springer-Verlag New York, 2005.

[33] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2328, 2000.

[34] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 2000.


INL
Idaho National Laboratory