INL/EXT-15-36649

Light Water Reactor Sustainability Program

Reduced Order Model Implementation in the Risk-Informed Safety Margin Characterization Toolkit

September 2015

DOE Office of Nuclear Energy

DISCLAIMER

This information was prepared as an account of work sponsored by an agency of the U.S. Government. Neither the U.S. Government nor any agency thereof, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness, of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, do not necessarily constitute or imply its endorsement, recommendation, or favoring by the U.S. Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the U.S. Government or any agency thereof.

INL/EXT-15-36649

Light Water Reactor Sustainability Program

Reduced Order Model Implementation in the Risk-Informed Safety Margin Characterization Toolkit

D. Mandelli, C. Smith, A. Alfonsi, C. Rabiti, J. Cogliati, H. Zhao,
I. Rinaldi, D. Maljovec, P.Talbot, B. Wang, V. Pascucci

September 2015

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov/lwrs

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

EXECUTIVE SUMMARY

The RISMC project aims to develop new advanced simulation-based tools to perform Computational Risk Analysis (CRA) for the existing fleet of U.S. nuclear power plants (NPPs). These tools numerically model not only the thermal-hydraulic behavior of the reactors primary and secondary systems, but also external event temporal evolution and component/system ageing. Thus, this is not only a multi-physics problem being addressed, but also a multi-scale problem (both spatial, µm-mm-m, and temporal, ms-s-minutes-years). As part of the RISMC CRA approach, a large amount of computationally-expensive simulation runs may be required. An important aspect is that even though computational power is growing, the overall computational cost of a RISMC analysis using brute-force methods may be not viable for certain cases. A solution that is being evaluated to assist the computational issue is the use of reduced order modeling techniques. During the FY2015, we investigated and applied reduced order modeling techniques to decrease the RISMC analysis computational cost by decreasing the number of simulation runs; for this analysis improvement we used surrogate models instead of the actual simulation codes. This report focuses on the use of reduced order modeling techniques that can be applied to RISMC analyses in order to generate, analyze, and visualize data. In particular, we focus on surrogate models that approximate the simulation results but in a much faster time (microseconds instead of hours/days). We apply reduced order and surrogate modeling techniques to several RISMC-types of analyses using RAVEN and RELAP-7 and show the advantages that can be gained.

CONTENTS

FIGURES

TABLES

ACRONYMS

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

PWR	Pressurized Water Reactor
R&D	Research and Development
RAVEN	Risk Analysis in a Virtual Environment
RISMC	Risk-Informed Safety Margin Characterization
ROM	Reduced Order Model
RPV	Reactor Pressure Vessel
SAMG	Severe Accident Management Guideline
SVM	Support Vector Machine
SVR	Support Vector Regressor
T-H	Thermal-Hydraulics

Reduced Order Model Implementation in the Risk-Informed Safety Margin Characterization Toolkit

1. INTRODUCTION

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

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

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

The risk simulation contains all deterministic elements that impact accident evolution such as:

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

In addition to stochastic ones such as:

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

The stochastic analysis [3] is performed in two steps:

1. Sampling the stochastic parameters, and
2. Evaluating the system response for the given set of sampled parameters

In the RISMC applications, system simulator codes model not only plant thermo-hydraulic, thermo-mechanic, neutronics, and ageing behavior but also model external event and human interactions with the plant itself. This is not only a *multi-physics* problem (i.e., different sets of equations are solved) but also a *multi-scale* one (i.e., both temporal and spatial scales). The drawback is that a single plant accident analysis (e.g., prediction of the seismic response of a BWR that underwent to a 60 years life extension license) might require long computational time that grows exponentially if multiple runs (through the sampling process) are needed.

In [3] we have focused our attention on the sampling strategies that we are employing. We initially employed classical sampling algorithms like Monte-Carlo, Grid and Latin Hypercube. In addition, we investigated more advanced sampling algorithms that aim to reduce the number of samples required to perform the desired stochastic analysis. We have shown how this reduction would allow the user to greatly reduce the computational costs of a typical RISMC analysis.

In this report, however we are focusing on how we can reduce the computational costs by more broadly employing Reduced Order Modeling techniques in typical RISMC-type analyses. We will show how reduced order modeling techniques can be applied to any RISMC analysis to generate, analyze and visualize data. In particular, we focus on surrogate models that approximate the simulation results but in a much faster time (microseconds instead of hours/days). We apply reduced order and surrogate modeling techniques to several RISMC types of analyses using RAVEN [4] and RELAP-7 [5] and show the advantages that can be gained.

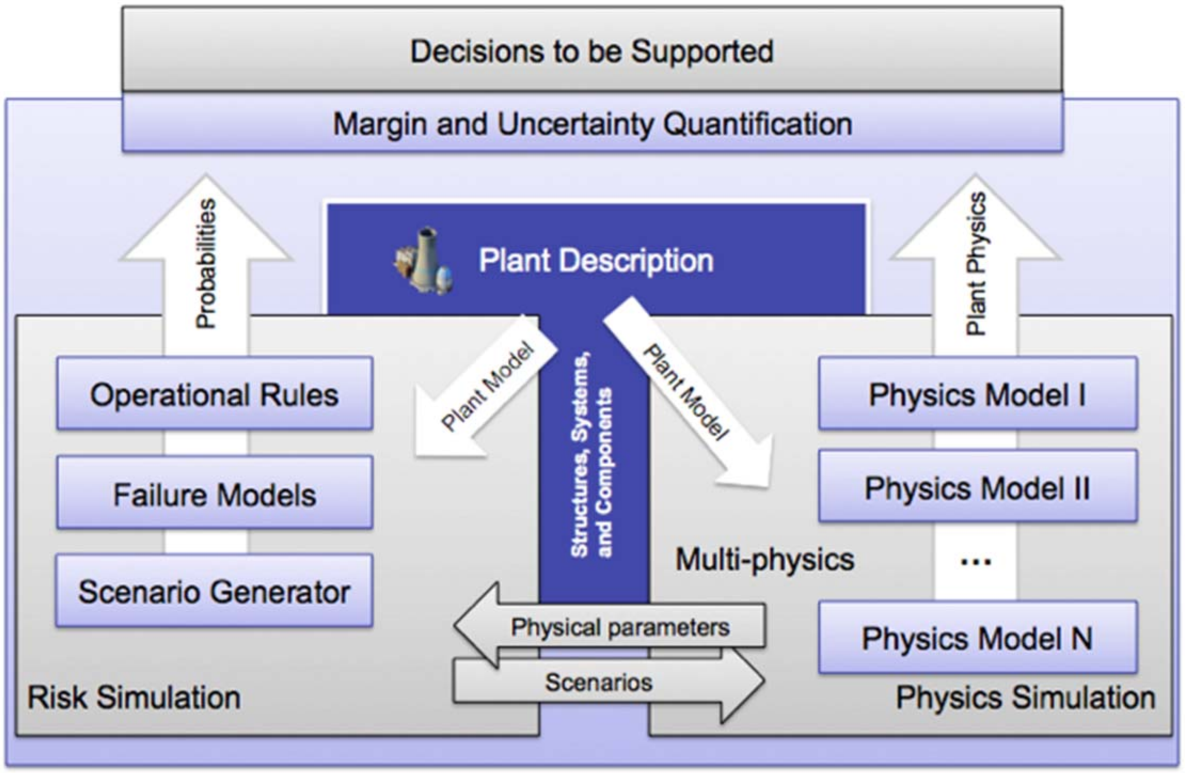


Figure 1 – The approach used to support RISMC analysis

The report is structured in three main parts that are described as follows:

Part 1: RISMC Approach

- Section 2 (page 1): Description of the RISMC PRA approach
- Section 3 (page 6): Description of the RISMC toolkit: RAVEN, CROW and RELAP-7
- Section 4 (page 11): Overview of Reduced Order Modeling and its applications in the RISMC analysis steps
- Section 5 (page 18): Summary of analytical tests that will be employed in Sections 9 and 10

Part 2: Reduced Order Modeling

- Section 6 (page 25): Summary and examples of how reduced order modeling techniques can be employed in RISMC-types of applications
- Section 7 (page 39): Description of temporal ROM
- Section 8 (page 44): Description of the surrogate models available in RAVEN and focus on the ones employed in this report
- Section 9 (page 50): Classifiers comparison and convergence tests
- Section 10 (page 71): Regressor models comparison and convergence tests

Part 3: Conclusions

- Section 11 (page 106): Conclusions and summary of the work presented

2. RISMC APPROACH

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

- Modeling of the thermal-hydraulic behavior of the plant [6]
- Modeling of external events such as flooding [7]
- Modeling of the operators responses to the accident scenario [8]

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

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

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

- Uncertain parameters
- Stochastic failure of system/components

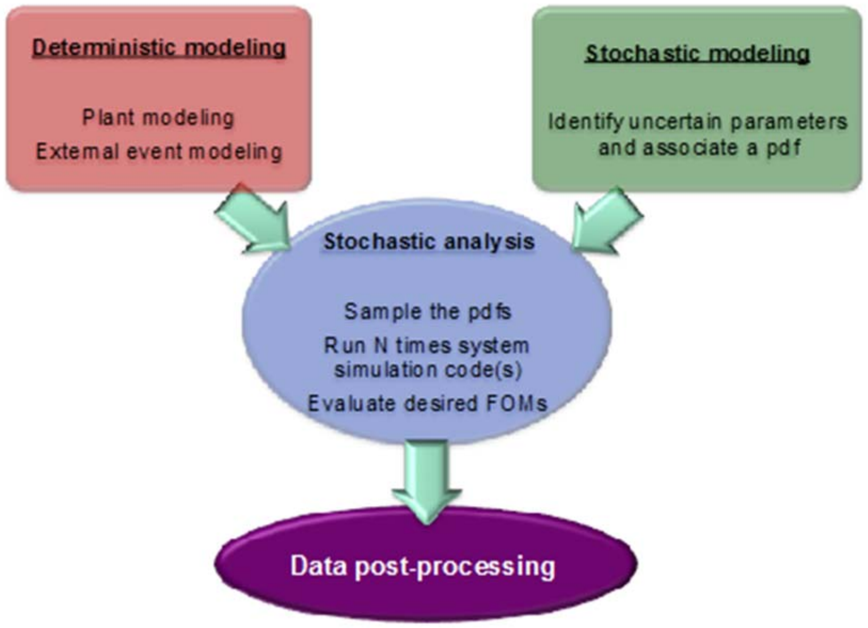


Figure 2 – Overview of the RISMC modeling approach

The RISMC approach heavily relies on multi-physics system simulator codes (e.g., RELAP-7 [5]) coupled with stochastic analysis tools (e.g., RAVEN [4]). From a mathematical point of view, a single simulator run can be represented as a single trajectory in the phase space. The evolution of such a trajectory in the phase space can be described as follows:

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

where:

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

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

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

3. RISMC TOOLKIT

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

- RELAP-7 [5] (see Section 2.2): the code responsible for simulating the thermal-hydraulic dynamics of the plant.

- RAVEN [4,10] (see Section 6): has two main functions: 1) act as a controller of the RELAP-7 simulation and 2) generate multiple scenarios (i.e., a sampler) by stochastically changing the order and/or timing of events.

- PEACOCK [11] (see Section 2.3): the Graphical User Interface (GUI) that allows the user to create/modify input files of both RAVEN and RELAP-7. Also, it monitors the simulation in real time while it is running.

- GRIZZLY [12]: the code that simulates the thermal-mechanical behavior of components in order to model component aging and degradation. Note that for the analysis described in this report, aging was not considered.

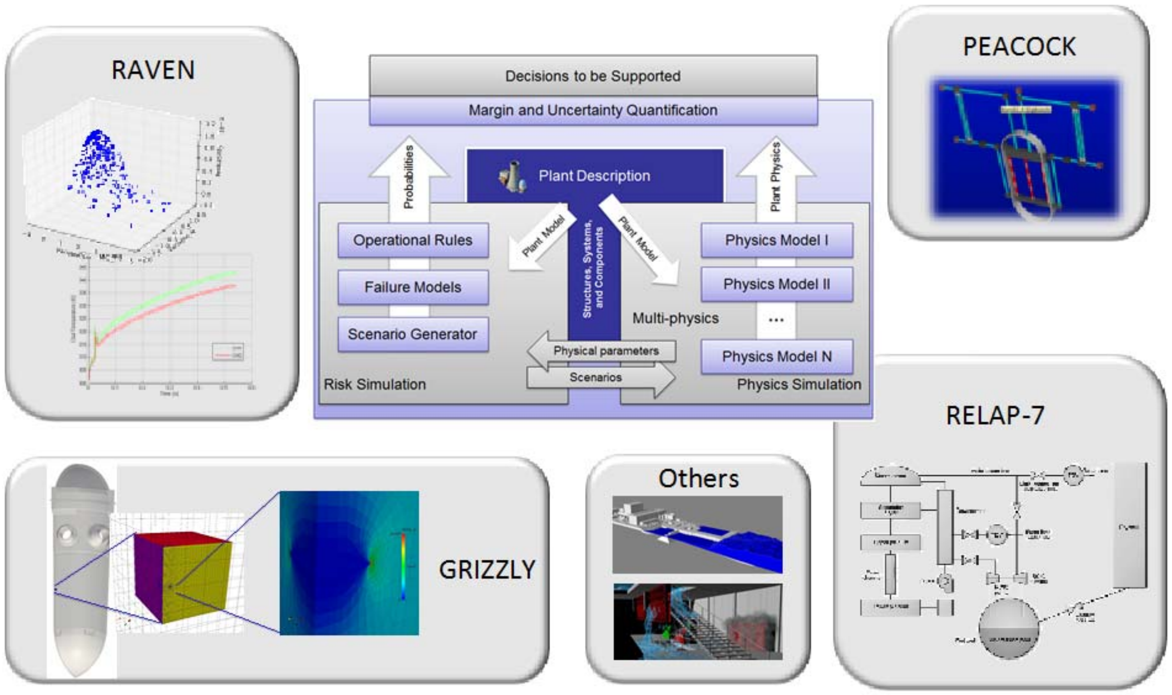


Figure 3 – Overview of the RISMC toolkit

3.1 RELAP-7

The RELAP-7 code [5] is the new nuclear reactor system safety analysis code being developed at the Idaho National Laboratory (INL). RELAP-7 is designed to be the main reactor system simulation toolkit for the RISMC Pathway. The RELAP-7 code development is taking advantage of the progress made in the past several decades to achieve simultaneous advancement of physical models, numerical methods, and software design. RELAP-7 uses the INLs MOOSE (Multi-Physics Object-Oriented Simulation Environment) framework [9] for solving computational engineering problems in a well-planned, managed, and coordinated way. This allows RELAP-7 development to focus strictly on systems analysis-type physical modeling and gives priority to retention and extension of RELAP5s multidimensional system capabilities.

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

3.2 Risk Analysis in a Virtual ENviroment (RAVEN)

RAVEN (Risk Analysis in a Virtual ENviroment) [4,10] is a software framework that acts as the control logic driver for the thermal-hydraulic code RELAP-7, a newly developed software at INL. RAVEN is also a multi-purpose Probabilistic Risk Assessment (PRA) code that allows for probabilistic analysis of complex systems. It is designed to derive and actuate the control logic required to simulate both plant control system and operator actions (guided procedures) and to perform both Monte-Carlo sampling [13] of random distributed events and dynamic branching-type [14] based analysis.

RAVEN consists of two main software components:

1. Simulation controller (see Section 3.2.1)
2. Statistical framework (see Section 3.2.2)

3.2.1 Simulation controller

One task of RAVEN is to act as controller of the RELAP-7 simulation while simulation is running. This control action is performed by using two sets of variables [10]:

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

The manipulation of these two data sets of variables is performed by two components of the RAVEN simulation controller (see Figure 4):

- RAVEN control logic: is the actual system control logic of the simulation where, based on the status of the system (i.e., monitored variables), it updates the status/value of the controlled parameters

- RAVEN/RELAP-7 interface: is in charge of updating and retrieving RELAP-7/MOOSE component variables according to the control logic

A third set of variables, i.e. *auxiliary variables*, allows the user to define simulation specific variables that may be needed to control the simulation. From a mathematical point of view, auxiliary variables are the ones that guarantee the system to be Markovian [15], i.e., the system status at time $t = \bar{t} + \Delta t$ can be numerically solved given only the system status at time $t = \bar{t}$.

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

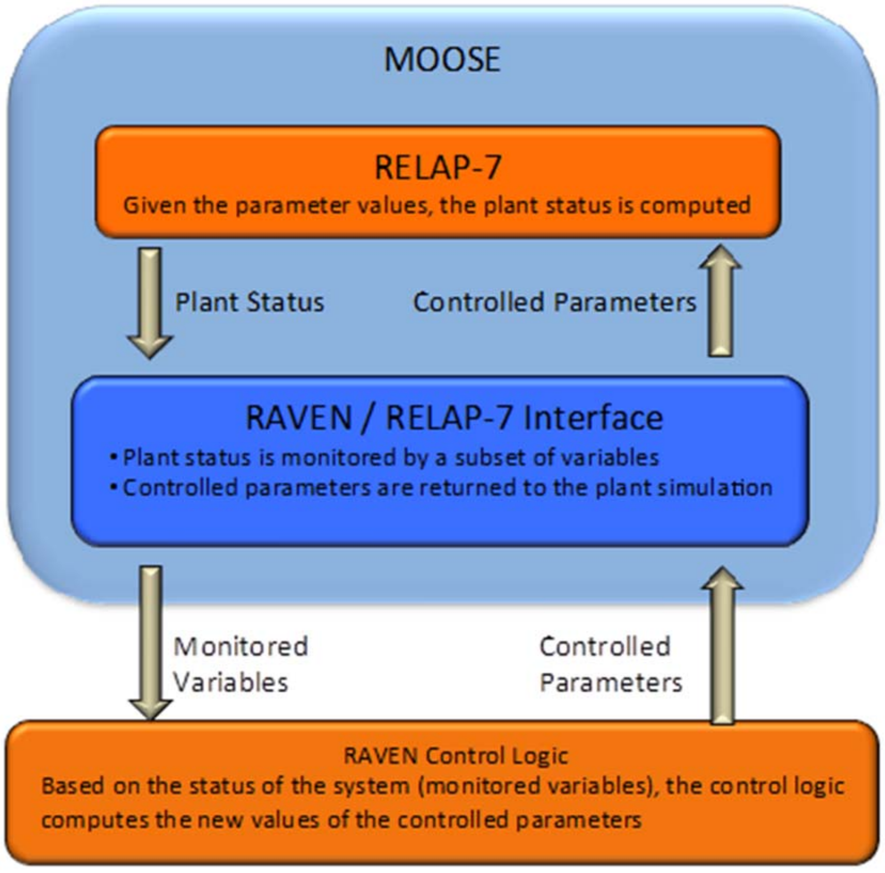

MOOSE
RELAP-7
Given the parameter values, the plant status is computed
Plant Status
Controlled Parameters
RAVEN / RELAP-7 Interface
• Plant status is monitored by a subset of variables
• Controlled parameters are returned to the plant simulation
Monitored Variables
Controlled Parameters
RAVEN Control Logic
Based on the status of the system (monitored variables), the control logic computes the new values of the controlled parameters

Figure 4 – RAVEN simulation controller scheme

3.2.2 RAVEN Statistical Framework

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

- Sampling of codes: either stochastic (e.g., Monte-Carlo [16] and Latin Hypercube Sampling [17]) or deterministic (e.g., grid and Dynamic Event Tree [18])
- Generation of Reduced Order Models [19] also known as Surrogate models
- Post-processing of the sampled data and generation of statistical parameters (e.g., mean, variance, covariance matrix)

Figure 5 shows an overview of the elements that comprise the RAVEN statistical framework:

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

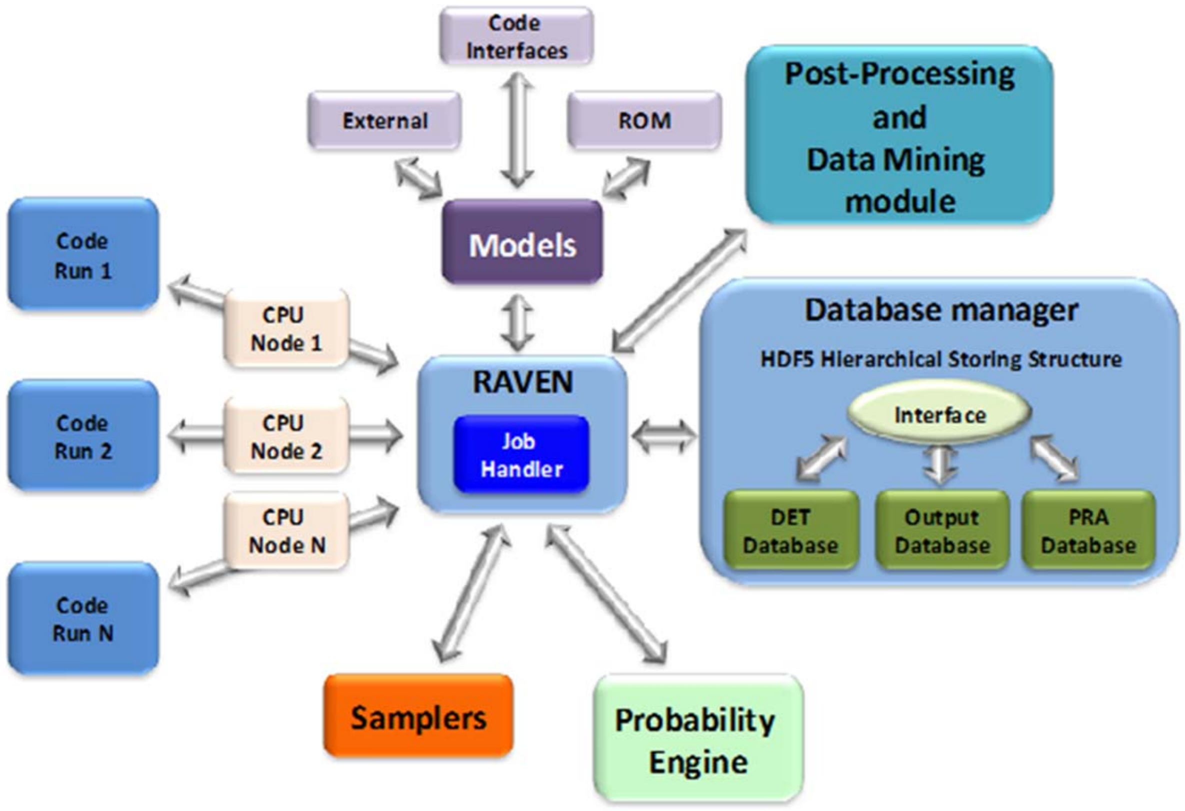


Figure 5 – Scheme of RAVEN statistical framework components

3.2.3 CROW Library

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

- Interface with BOOST [20]. BOOST[1] is a set of libraries for the `C++` programming language that provide support for tasks and structures such as linear algebra, pseudorandom number generation, multithreading, image processing, regular expressions, and unit testing. For our applications we are using the mathematical/statistical[2] library of BOOST which contains a wide selection of univariate statistical distributions and functions that operate on them (pdf and cdf calculation along with random number generation).

- Multi-variate distributions. This module contains the same functions mentioned above (pdf and cdf calculation along with random number generation) but applied to multi-variate distributions. Not only multi-variate normal distributions are modeled but also generic multi-variate distributions defined over a set of samples scattered or grid distributed.

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

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

4. REDUCED ORDER MODELING

Reduced order modeling is a fairly generic term. Its semantics changes from field to field (e.g., computer science, engineering and so on.) For the scope of this report, we include in the Reduced Order Modeling term all methodologies and algorithms that aim to reduce the complexity of a problem, where "a problem" is considered a broad term and can be either an abstract entity (e.g., a simulator code or a dataset) or a concrete entity (e.g., an experimental facility, a power plant).

In the RISMC applications, we are mainly dealing with numerical entities such as system codes like RELAP-7. It is relevant to highlight that the safety modeling of a nuclear system is not only a thermal-hydraulic problem but several other models are required: neutron transport, thermo-mechanics, chemistry, fracture propagation, etc. Note how the overall problem is not only multi-physics but also multi-scale both in the spatial scale but also in the temporal scale. While the coupling of these processes can be implicitly solved numerically, the RISMC project is focusing its attention toward the use of reduced order modeling techniques in order to decrease the computational cost (in terms of both computing power and memory requirement.)

For the scope of this report we divided the concept of Reduced Order Modeling into three main categories:

- *Reduced physics*: use of simulator codes that employ simplified physics problems. An example is the use of diffusion codes to solve neutronic problems instead of transport codes. In this category we also include the possibility to use in the same simulation run high and low fidelity models depending on the boundary conditions of the simulation.

- *Reduced dimensionality*: a simulation run can be seen as a trajectory in the phase space and a single point in the input space. The dimensionality of these spaces can be very high for the complex analyses. This category includes all methods than aim to reduce the dimensionality of these spaces and project the original problem into the reduced space.

- *Surrogate model*: surrogate models are mathematical objects that emulate the behavior of a code by learning its input/output relations and reconstructing such relations through a regression/interpolation based approach.

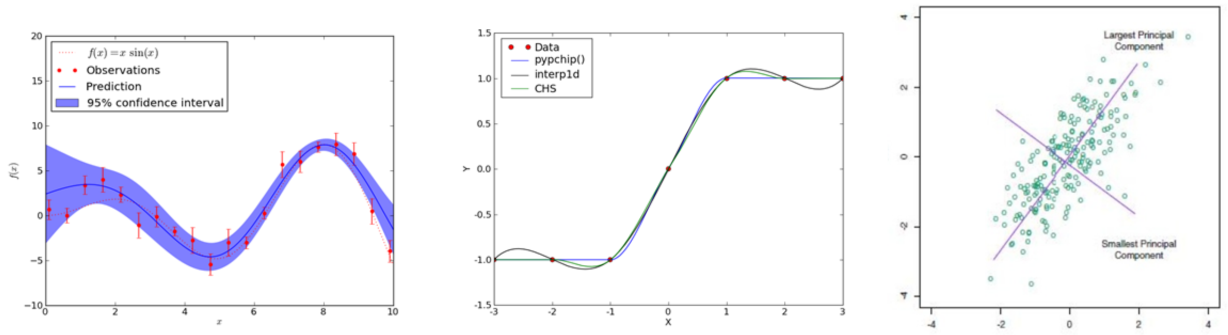


Figure 6 – Examples of Reduced Order Modeling: regression (left), interpolation (center) and dimensionality reduction (right) (source: http://scikit-learn.org)

For the last two categories the set of methodologies employed are typically based on regression (e.g., Gaussian process models [21]), interpolation (e.g., spline kernel and linear kernel) and dimensionality reduction algorithms (e.g., Principal Component Analysis –PCA– [22] and ISOMAP [23]). Figure 6 illustrates all the three algorithms above mentioned.

In Figure 2 we have shown the four basic steps of the RISMC approach. In order to illustrate how Reduced Order Modeling techniques can be applied in the RISMC approach, we have indicated in Figure 7 the set of methods that can be applied to each of the four RISMC steps.

- **Deterministic modeling:** employment of reduced physics codes (i.e., multi-fidelity codes) or surrogate models instead of the actual codes

- **Stochastic modeling:** reduction of the number of stochastic parameters to be sampled (i.e., reduction of the dimensionality of the input space)

- **Stochastic Analysis:** reduction of the number of simulations to run by carefully choosing a minimum set of simulations that maximize the amount of information required by the analysis (adaptive – smart – sampling)

- **Data Post-Processing:** use of stochastic analysis tools (e.g., Kernel Density Estimation methods) to summarize large amounts of data and employment of advanced topology-based visualization tools to visualize high dimensional data.

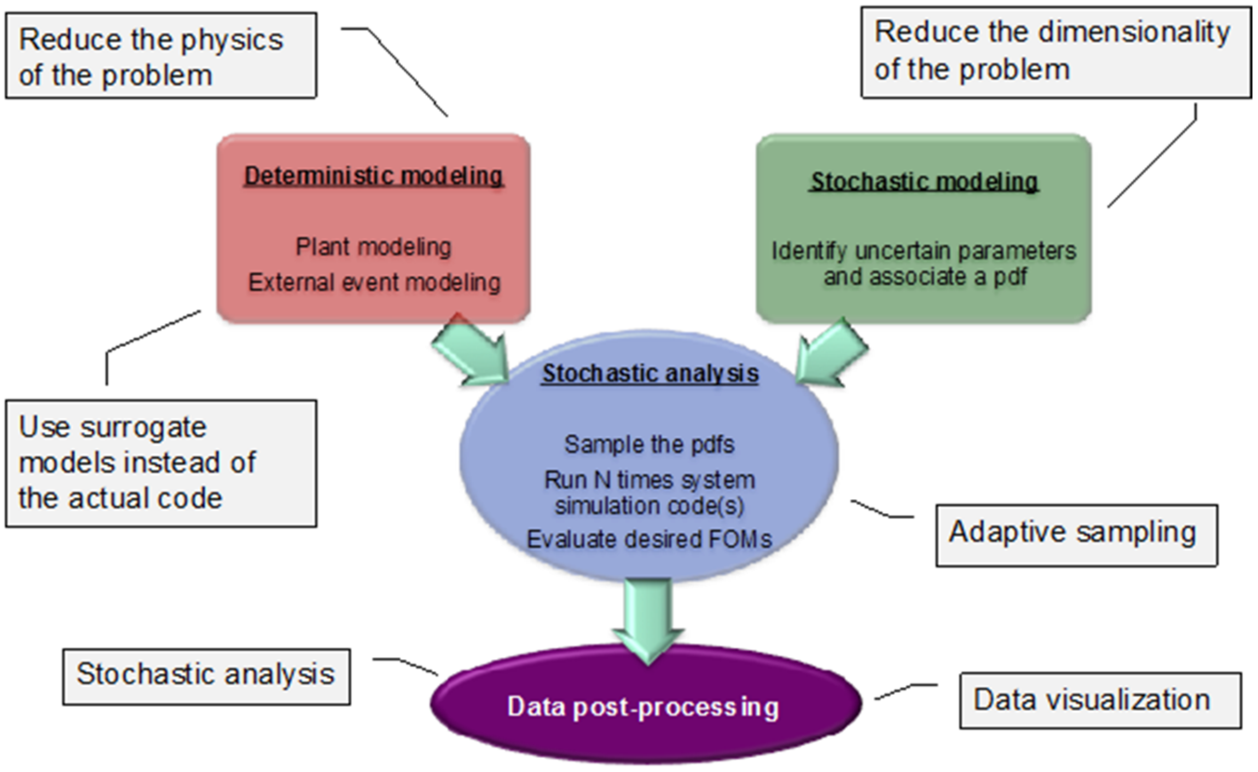


Figure 7 – Overview of how Reduced Order Modeling can be applied to the four main steps of a typical RISMC analysis

In the present report we will focus in great detail on surrogate models and adaptive sampling techniques. We will briefly describe the dimensionality reduction topic (see Section 4.2) and the visualization of the obtained data (see Section 6.3) since they are under development within in the RAVEN statistical framework. We will not cover the concept of physics reduction (i.e., multi-fidelity codes).

4.1 Surrogate Models

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

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

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

$$\boldsymbol{\Theta}(\boldsymbol{s}): \boldsymbol{s_i} \to \boldsymbol{\Theta}(\boldsymbol{s_i}) \cong \boldsymbol{\mathcal{H}}(\boldsymbol{s_i}) \qquad \text{Eq. 2}$$

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

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

In model based ROMs the prediction is performed using a blend of interpolation and regression algorithms[3]. Examples are:

- Gaussian Process Models (GPMs) [21]
- Multi-dimensional spline interpolators [24]

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

[3] Interpolation: Given a set of N data points (x_i, y_i) $i = 1, \dots, N$, interpolation aim to find a function F that is of some user-defined form (e.g., linear) that has the values in that points exactly as specified, i.e., it satisfies $F(x_i) = y_i$.
Given the same set of N data points (x_i, y_i) $i = 1, \dots, N$, regression, regression look for a function that minimizes some cost, usually sum of squares of errors $\sum_{i=1}^{n}(F(x_i) - y_i)^2$. The requirement $F(x_i) = y_i$ is usually not imposed.

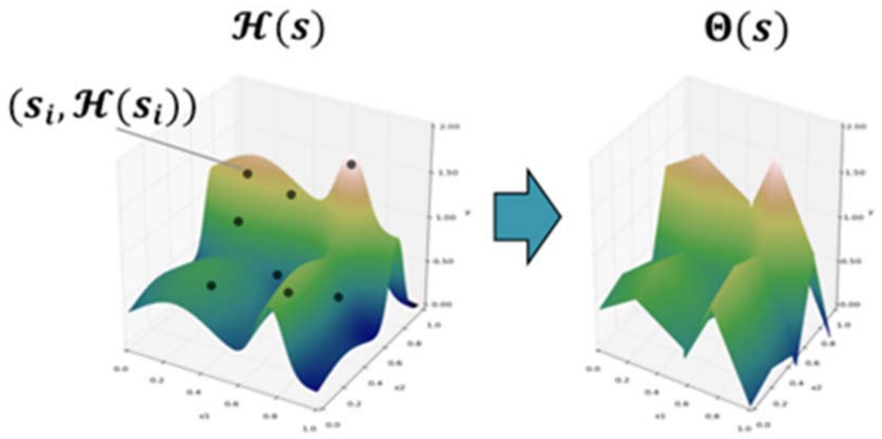


Figure 8 – Example of reduced order modeling approximation of a sampled 3-D response surface

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

- K Nearest Neighbor classifier (KNN) [25]
- Graph based models [26]

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

4.2 Dimensionality Reduction

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

$$\mathfrak{I}: \mathbb{R}^D \to \mathbb{R}^d \quad \text{where } d < D$$

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.

Linear algorithms, such as PCA [22] or multidimensional scaling (MDS) [27], have the advantage that they are easier to implement; however, they can only identify linear correlation among variables. On the other hand, methodologies such as Local Linear Embedding [28] and ISOMAP [23] are more computationally intensive but they are able to identify non-linear correlations. Figure 9 shows two examples of linear and non-linear correlations. In both cases points are distributed in a 2-dimensional space (i.e., characterized by 2 variables: x, y) but they are lying in a 1-dimensional space.

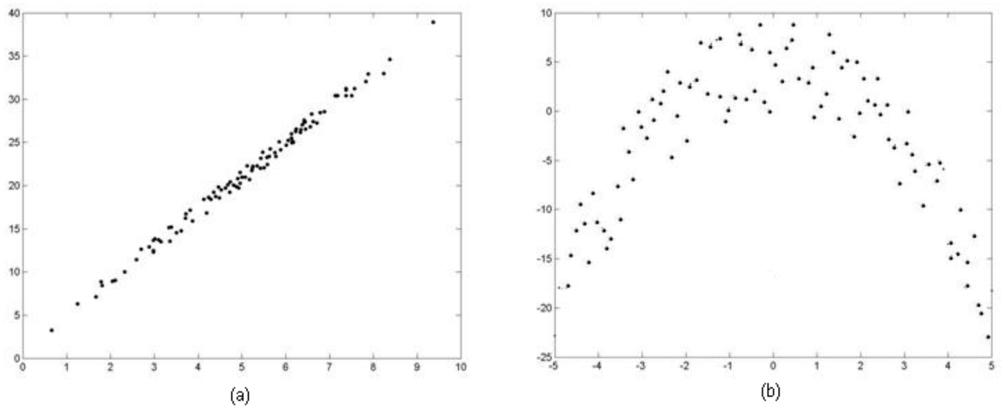


Figure 9 – Example of linear (a) and non-linear (b) correlation between 2 variables.

The main idea behind PCA [22] 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] Σ. 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.

Figure 10 shows an example of dimensionality reduction using PCA for a data set distributed in a 2-dimensional space. After performing the eigenvalue-eigenvector decomposition of the covariance matrix, the algorithm chooses the eigenvector having the largest eigenvalue (i.e., λ_1) as subspace to project the original data. The algorithm is very easy to implement but, on the other hand, PCA is not able to identify non-linear correlations of more complex data sets.

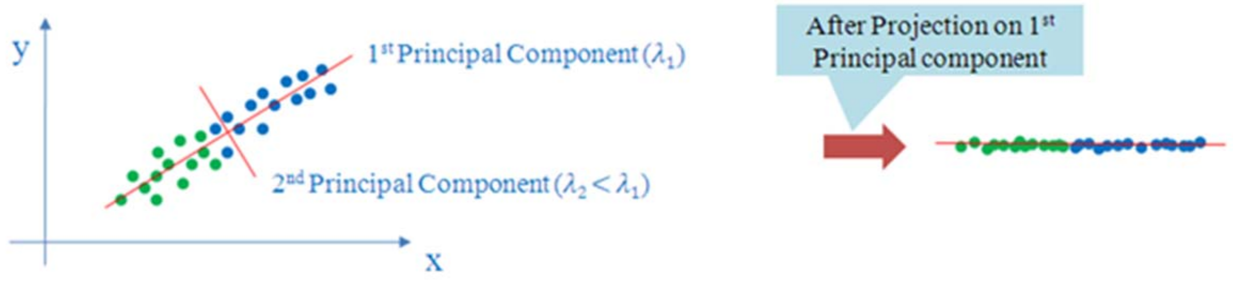

y
1st Principal Component (λ_1)
2nd Principal Component ($\lambda_2 < \lambda_1$)
x
After Projection on 1st Principal component

Figure 10 – Example of dimensionality (from *D* = 2 to *d* = 1) reduction using PCA

[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 Σ is determined as $\Sigma = \frac{1}{\Lambda-1} Z\,Z'$

.

Multidimensional scaling MDS [27] 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.

Since PCA and MDS are linear algorithms and, thus, limited to the data sets which include only linear correlations among variables it was decided to consider non-linear algorithms such as those based on manifold analysis. In particular, the ISOMAP[5] [23] algorithm has been considered. The ISOMAP algorithm provides a simple method for estimating the intrinsic geometry of a data manifold based on a rough estimate of each data points neighbors on the manifold. ISOMAP is one representative of isometric mapping methods, and extends MDS by incorporating the geodesic distances[6] (distance along the manifold) imposed by a weighted graph. ISOMAP is distinguished by its use of the geodesic distance induced by a neighborhood graph embedded in the classical scaling. The algorithm is implemented by following these two steps:

1. Estimate the geodesic distance between points in inputs using shortest-path distances on the data sets k nearest neighbor[7]. The connectivity of each data point in the neighborhood graph is defined as its nearest k Euclidean neighbors in the high-dimensional space.

2. Use MDS to find points in low-dimensional Euclidean space whose interpoint distances match the distances found in Step 1.

4.3 Reduced Order Modeling in RAVEN: Workflow

As shown in Section 3.2, the RAVEN statistical framework is a flexible environment that allows the user to perform stochastic sampling of codes, analysis of the data generated and generation of Surrogate Models. In addition RAVEN is able to operate not only on end-user types of machines (i.e., desktop, laptop and workstations) but to perform parallel computing on High Performance Computing (HPC) such as FALCON located at INL.

For this report we have performed several stochastic analyses using RELAP-7 and external models (mainly test functions coded as python script). Since each RELAP-7 run may take several hours we have decided to perform the stochastic analysis that required RELAP-7 on FALCON while the rest of the analyses on a laptop computer.

Figure 11 gives an overview of the possible path that has been followed in this report using RAVEN and RELAP-7 on both end-user and HPC machines.

[5] http://isomap.stanford.edu/

[6] In graph theory, the distance between two vertices in a graph is the number of edges in a shortest path connecting them. This is also known as the geodesic distance.

[7] ISOMAP defines the geodesic distance to be the sum of edge weights along the shortest path between two nodes.

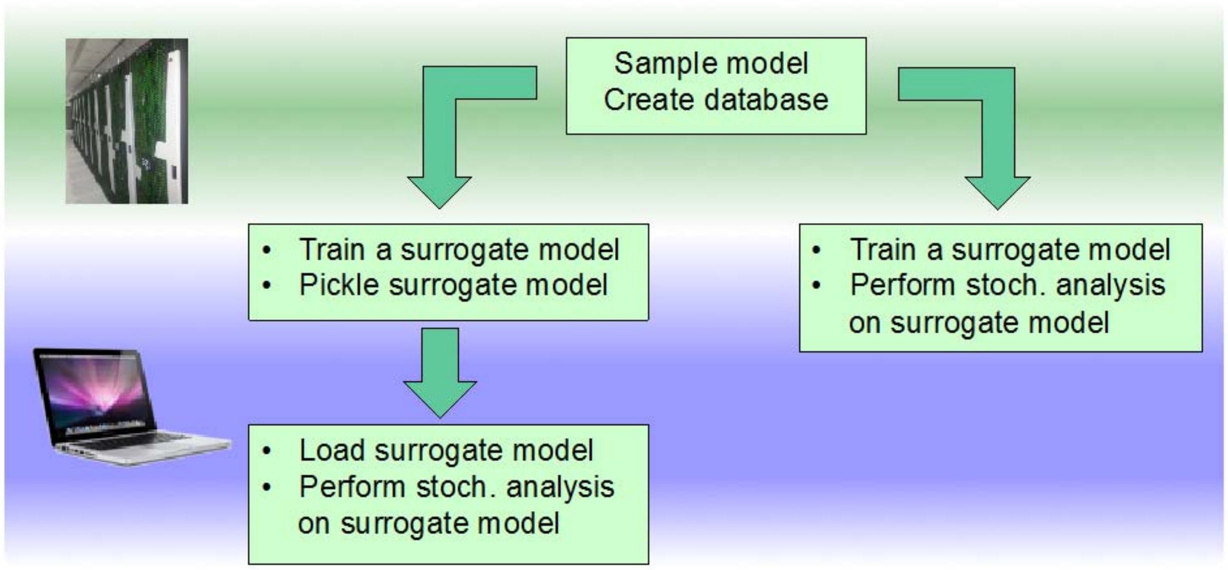

Sample model
Create database
• Train a surrogate model
• Pickle surrogate model
• Train a surrogate model
• Perform stoch. analysis on surrogate model
• Load surrogate model
• Perform stoch. analysis on surrogate model

Figure 11 – RAVEN Reduced Order Modeling workflow

5. MODELS

In this section we present the set of models that we used to test some of the surrogate models available in RAVEN. These models are both analytical (see Section 5.1) and also RISMC related models using RELAP-7 (see Section 5.2). In the next two sections these models are described in detail.

5.1 Analytical Models

5.1.1 Rosenbrock Function

The Rosenbrock function (see Figure 12) is a three dimensional surface that is determined by the following equation:

$$y1 = 2\ 10^{-3}\,((1 - x1)^2 + 100(x2 - x1^2)^2) + 2 \qquad \textbf{Eq. 3}$$

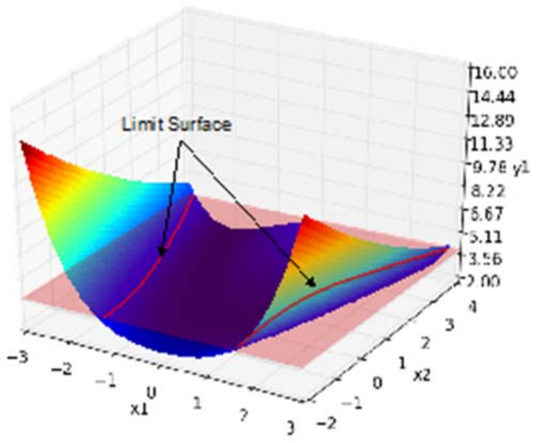


Figure 12 – Plot of the Rosenbrock function

The distinctiveness of this function is that it has three local maxima and a local minima region. This function is used for both classifier and regressor types of surrogate models. For the classifier case, we aim to identify the location of the points $(x1, x2)$ such that $y1 = 4.0$; such a region is called the *limit surface* [32] and it is shown in red in Figure 12.

5.1.2 Triangular Region

This test case is being used uniquely to test some of the classifiers available in RAVEN. This test has been chosen in order to test classifier performance to deal with closed discontinuous regions. This region is shown in Figure 13.

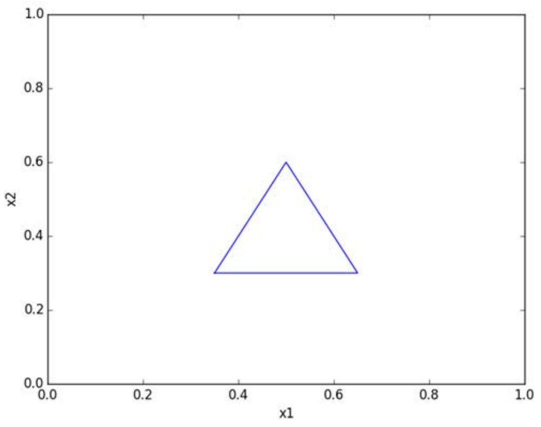


Figure 13 – Triangular region plot

5.1.3 Paraboloid function

The Paraboloid function (see Figure 14) is a three dimensional surface that is described by the following equation:

$$y1 = 3.0\ x1^2 + 2.0\ x2^2 \qquad \textbf{Eq. 4}$$

This test function is used to test some of the surrogate models (i.e., regressors) that will be used for uncertainty propagation and sensitivity analysis type of applications.

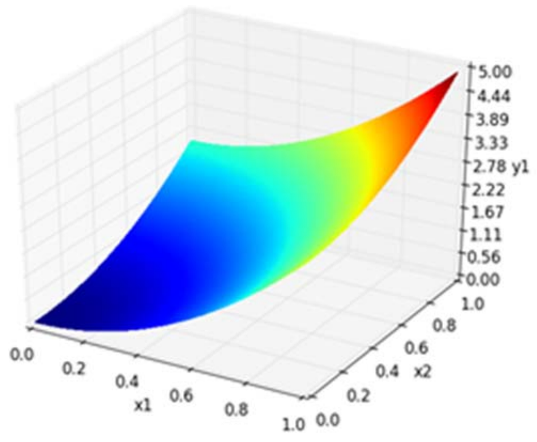


Figure 14 – Plot of the paraboloid function

5.2 RELAP-7 Models

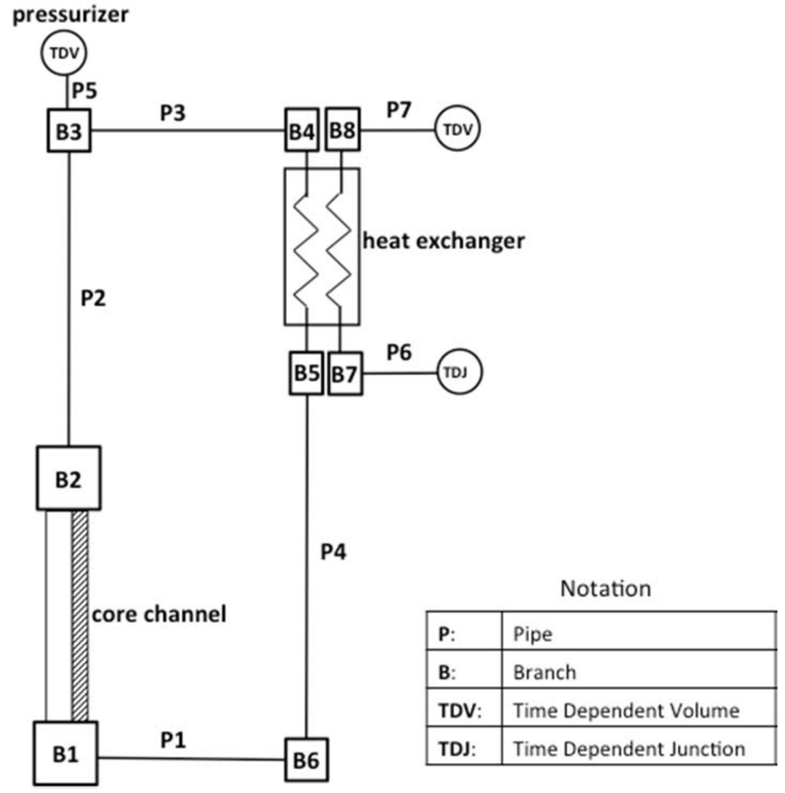
5.2.1 PWR Natural Circulation Loop

The model used in this report, as shown in Figure 15, is an imaginary scaled experiment [29]. The typical PWR core channel (2 × 2 fuel rods) is used for the heating section. The same fuel rod parameters are used as in the single channel numerical verification work. The cooler is a tube-shell type counter-current heat exchanger (HX) with primary coolant inside the pipe. The pipes are also arranged in a 2 × 2 square shape. The major TH parameters are listed in Table 1. The power is about 1% of the full power of the 4 fuel rods to simulate the typical decay heat.

The same wall friction and heat transfer correlations from RELAP5 [30] are used in RELAP-7 for the primary side. A very large number (106 W/m^2-K) is used for the HX secondary heat transfer coefficient to simulate boiling heat transfer. For this problem, we only focus on the primary side.

Table 1 – Major TH Parameters of the RELAP-7 PWR Natural Circulation Loop [29]

Parameter	Value
Heating power, W	3093.5
Initial pressure, MPa	15.5
Initial primary side temperature, K	559.15
Initial secondary side temperature, K	529.15
Initial primary side mass flow rate, kg/s	1.2935
Secondary side inlet flow velocity (TDJ), m/s	1.5
Secondary side inlet temperature (TDJ), K	529.15
Pipe diameters for P1, P2, P3, and P4, m	0.04
Pipe diameter for P5, m	0.01
Hydraulic diameter in core channel, m	0.01178
Length of the core channel, m	3.865
Inside diameter of HX pipe, m	0.01
Length of HX, m	4



Notation

P:	Pipe
B:	Branch
TDV:	Time Dependent Volume
TDJ:	Time Dependent Junction

Figure 15 – Scheme of the PWR natural circulation loop [29]

The transient starts with a uniform initial primary temperature and the given mass flow rate. The coast-down process then starts and is subsequently followed by gradually establishing natural circulation, which is driven by the temperature difference between the hot side (left) and cold side (right).

The PCT (Peak Clad Temperature) and the mass flow rate through the core channel are shown in Figure 16 and Figure 17, respectively. The evolution of the natural circulation can be divided into several stages. During the first stage (up to 4 s), the initial kinetic energy is rapidly damped by friction and form losses. The mass flow rate reaches the minimal value. During the second stage (4 s to 95 s), natural circulation is established and the PCT reaches the highest value at the end of this stage. The third stage (95 s to 700 s) shows the steady decreasing of PCT due to increased mass flow rate. When the cold column originating from the HX reaches the core, the PCT rapidly drops to the minimal value. During the fourth stage (700 s to 1200 s), the natural circulation capability is weakened due to decreasing hot section temperature. The stage ends when the last hot column originating from the core channel reaches the HX. After about half an hour, the transient almost approaches steady state. Except for the short duration during the coast-down stage, the flow is turbulent through the loop.

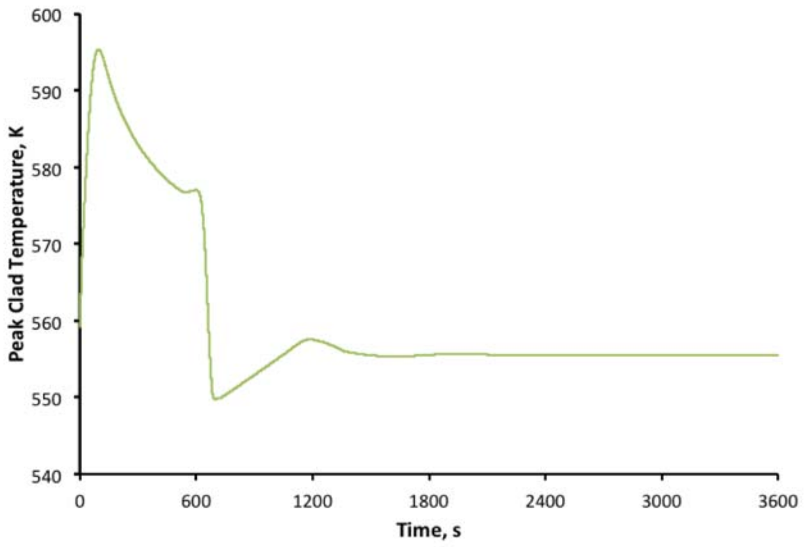


Figure 16 – PWR natural circulation loop: temporal transient for peak clad temperature [29]

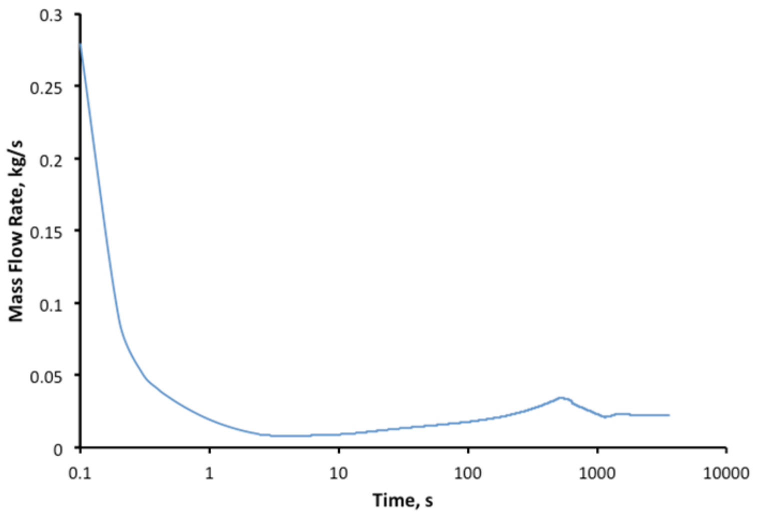


Figure 17 – PWR natural circulation loop: temporal transient for mass flow rate [29]

5.2.2 TMI PWR Model

A PWR simplified model has been set up based on the parameters specified in the OECD main steam line break (MSLB) benchmark problem [31]. The reference design for the OECD MSLB benchmark problem is derived from the reactor geometry and operational data of the TMI-1 Nuclear Power Plant (NPP), which is a 2772 MW two loop pressurized water reactor (see the system scheme shown in Figure 18).

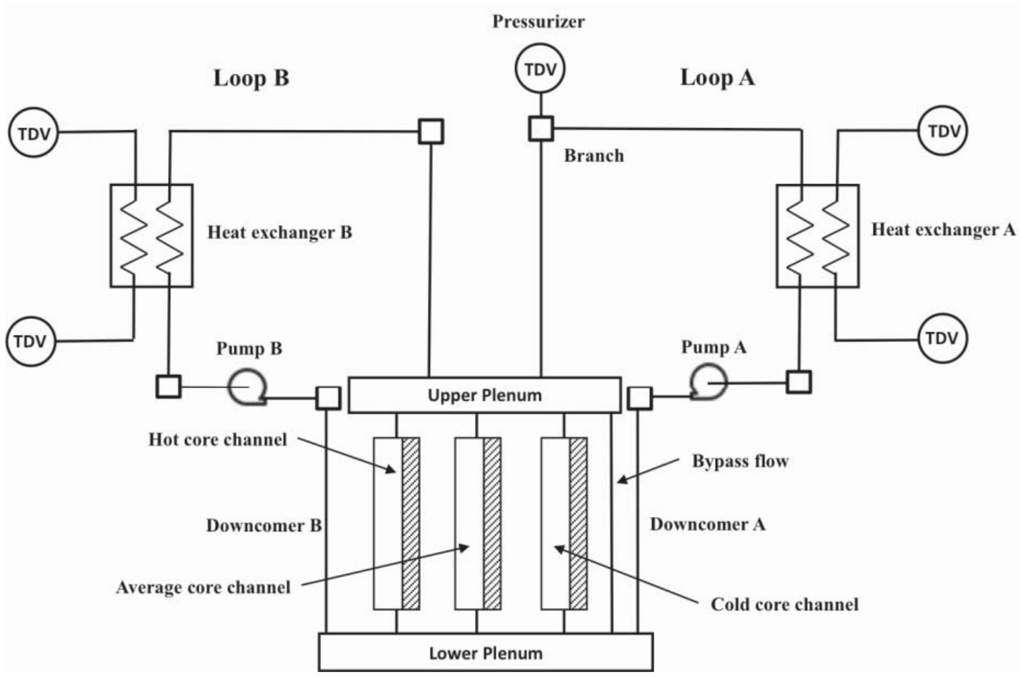


Figure 18 – Scheme of the RELAP-7 TMI loop

The scenario considered is a loss of off-site power (LOOP) initiating event caused by an earthquake followed by tsunami induced flooding. Depending on the wave height, it causes water to enter into the air intake of the DGs and temporary disable the DGs themselves. In more detail, the scenario is the following (see Figure 19):

1. An external event (i.e., earthquake) causes a LOOP due to damage of both off-site grid lines; the reactor successfully scrams and, thus, the power generated in the core follows the characteristic exponential decay curve

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

3. Due to a failure of the diesel generators, AC power is lost and conditions of SBO are reached; all core cooling systems are subsequently off-line (including the ECCS system)

4. Without the ability to cool the reactor core, its temperature starts to rise

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

6. When the AC power is recovered, the auxiliary cooling system (i.e., ECCS system) is able to cool the reactor core and, thus, core temperature decreases

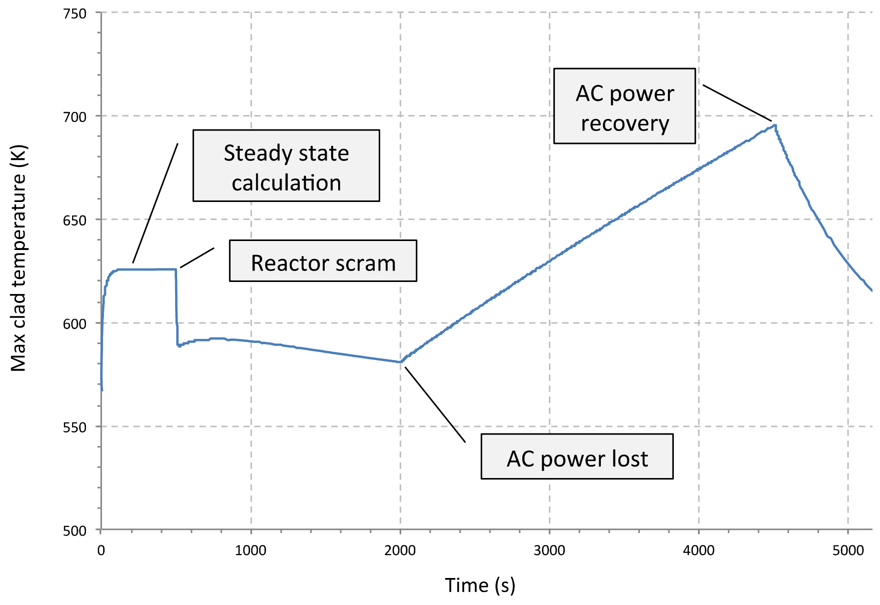


Figure 19 – TMI test case: Example of LOOP+SBO+AC recovery transient

6. APPLICATIONS

In this section we cover a set of applications that employs Reduced Order Modeling techniques. We will show a set of applications that are of interest in the RISMC pathway. In particular, we focus on a subset of the ones presented in Figure 7.

6.1 PRA Applications: Adaptive Sampling

The general adaptive sampling pipeline [45] begins by selecting some initial training data, running the simulation and obtaining a collection of true responses at these data points. Second, it fits a response surface surrogate model from the initial set of training data. Third, a set of candidate points is chosen in the parameter space based on certain sampling techniques, and the surrogate model is evaluated at these points, obtaining a set of approximated values. Fourth, each candidate point is assigned a score based on some adaptive sampling scoring function (usually derived from qualitative or quantitative relations between the training points, their true and estimated response values). Finally, the candidate(s) with the highest score(s) are selected and added to the set of training data to begin a new cycle.

As mentioned earlier this kind of sampling strategy requires not only simulator codes but also one, or possibly more, ROMs [19]. In our case, it is possible to view the code as a black-box $\boldsymbol{\mathcal{H}}$ that produces a set of output variables $\boldsymbol{y}$ given a set of input parameters $\boldsymbol{s}$:

$$\boldsymbol{\mathcal{H}}: \boldsymbol{s} \rightarrow \boldsymbol{y}(t) = \boldsymbol{\mathcal{H}}(\boldsymbol{\theta}, \boldsymbol{s}, t) \qquad \text{Eq. 5}$$

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

The main adaptive sampling steps are explained as follows (see Figure 20):

1. Perform a set of runs of the simulator code: the number of required runs may depend on the dimensionality of the input space.
2. Given the set of simulation runs obtained in Step 1, create a ROM. The objective of this ROM is to:
   - Infer the response of the simulator code, i.e., create an approximate output given the same set of input parameters
   - Predict the regions in the input space that maximizes the objective function
3. Employ the ROM to approximate the structure of the goal function
4. Identify a set of points that satisfy the conditions specified in the goal function and choose a subset of points from the ones obtained in Step 4 that maximize the goal function
5. Perform a simulation run for each of the points obtained in Step 5 using the simulator code
6. Repeat Steps 2 through 6 until convergence is reached

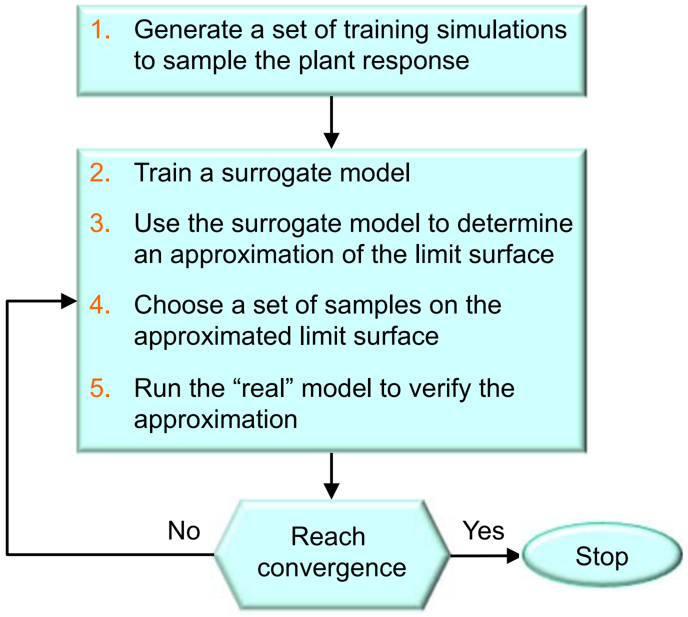

1. Generate a set of training simulations to sample the plant response
2. Train a surrogate model
3. Use the surrogate model to determine an approximation of the limit surface
4. Choose a set of samples on the approximated limit surface
5. Run the "real" model to verify the approximation
No
Reach convergence
Yes
Stop

Figure 20 – Workflow of adaptive sampling algorithms

As part of safety margin quantification, the RISMC approach aims to evaluate a set of limit surfaces [32]. A limit surface represents the boundaries in the input space (i.e., d-dimensional space; each dimension is one the d sampled variables) that separate the failure region (i.e., characterized by the undesired simulation outcome; e.g., core damage) from the success region (i.e., characterized by the desired simulation outcome; e.g., max clad temperature below 2200 F).

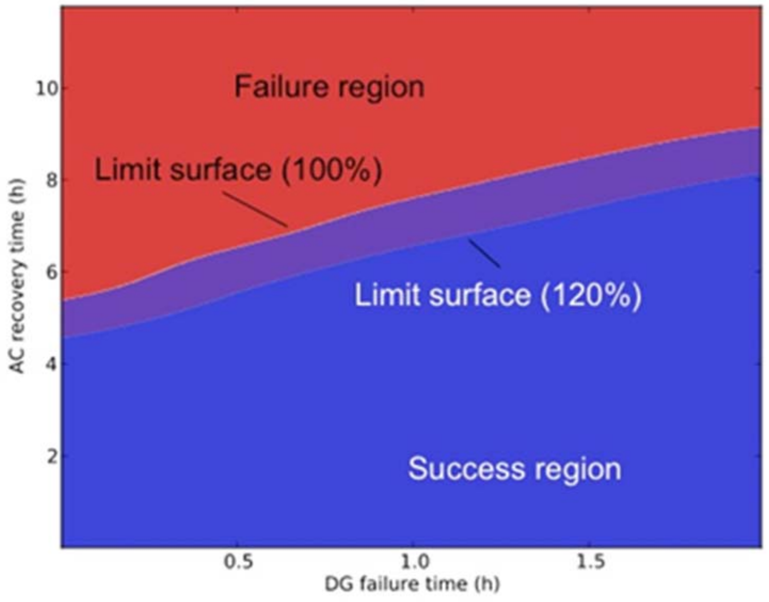


Figure 21 – Example of limit surface calculation for two different values of core power levels [33]

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

$$P_{CD} = \int_{\substack{failure \\ region}} pdf(\varpi)\, d\varpi \qquad \text{Eq. 6}$$

Equation 6 shows that P_{CD} is equal to the area of the failure region weighted by the probability of being in the failure region itself (through the probability distribution function $pdf(\varpi)$).

Figure 21 shows the limit surface in a 2-dimensional space generated in [33] using RAVEN coupled with RELAP-7 for a boiling water reactor station blackout initiating event. As part of the analysis, we were interested in the evaluation of the safety impacts of power uprate (reactor core power increased from 100 to 120%). Such evaluation has been performed by evaluating both the increased core damage probability ΔP_{CD} and the limit surface for both 100 and 120% reactor core power levels Note that ΔP_{CD} can be written as the same integral indicated in Eq. 6 but evaluated only in the *expanded failure region* ($\Delta\Omega_{Failure}$)

$$\Delta P_{CD} = \int_{\Delta\Omega_{Failure}} pdf(\varpi)\, d\varpi \qquad \text{Eq. 7}$$

6.1.1 RELAP-7 BWR SBO Test Case

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

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

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

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

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

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

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

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

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

2. Recovery time of DGs

The RELAP-7 PWR model has been set up based on the parameters specified in the OECD main steam line break (MSLB) benchmark problem [31]. The reference design for the OECD MSLB benchmark problem is derived from the reactor geometry and operational data of the TMI-1 Nuclear Power Plant (NPP), which is a 2772 MW two loop pressurized water reactor (see the system scheme shown in Figure 18).

For the scope of this report we wanted to show that one of the capabilities of RAVEN is to generate ROM and perform statistical analysis on them. For this case we collected the actual simulated data by RELAP-7 in [33], generated a ROM from such data and performed adaptive sampling on the ROM instead of the RELAP-7 code. In more detail, we performed the following steps:

1. Retrieved the hdf5 data generated by sampling RELAP-7 in [33]

2. Trained a ROM given the data retrieved in Step 1

3. Sampled on a 2-dimensional Cartesian grid the ROM obtained in Step 2

4. Performed adaptive sampling and limit-surface search

6.1.2 PWR SBO Limit surfaces

We performed the adaptive sampling analysis for this test case following an initial 6×6 Cartesian grid sampling for training. The sample locations and the estimated limit surface are shown for different steps of the sampling process, i.e. at iteration 1, 10, 30 and 60 (Figure 22) and 100, 150 and 185 (Figure 23) past the training sampling. For each iteration note how the sample locations are quickly approaching the exact location of the limit surface and the estimated limit surface is converging.

Table 2 compares the number of samples required to evaluate this limit surface by using classical Monte-Carlo and adaptive sampling. As can be seen in the table, a dramatic reduction in samples is needed when using adaptive sampling for this problem.

Table 2 – Number of samples required to evaluate the RELAP-7 PWR SBO limit surface by using classical Monte-Carlo and adaptive sampling (convergence in value is equal to $5x10^{-5}$)

	Number of Samples
Monte-Carlo	~10^7
Adaptive	185

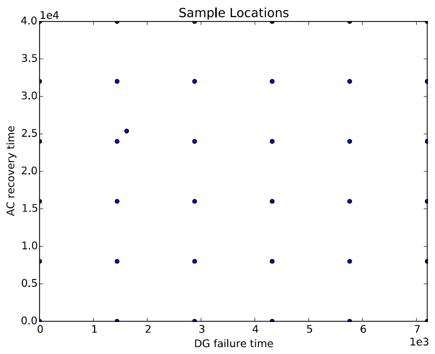
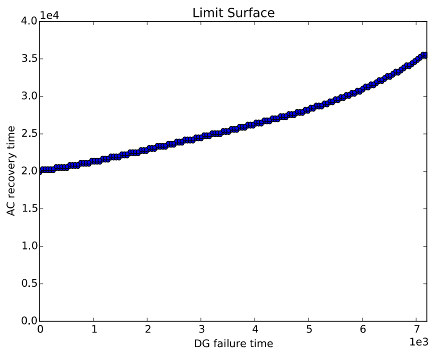
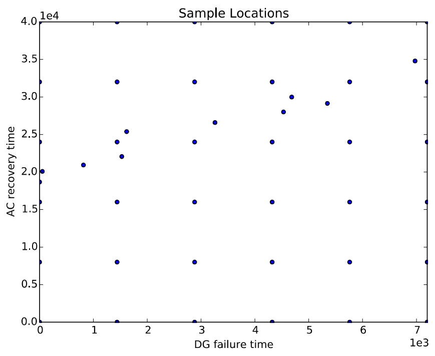
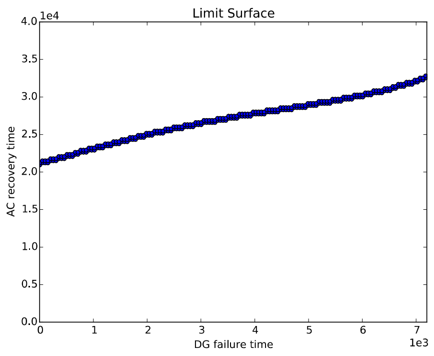
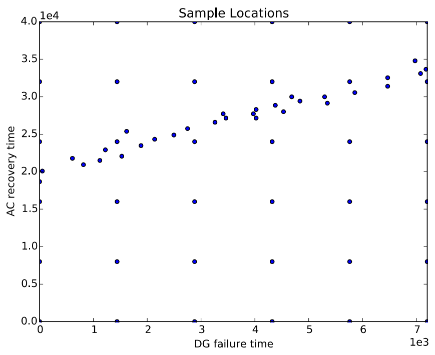
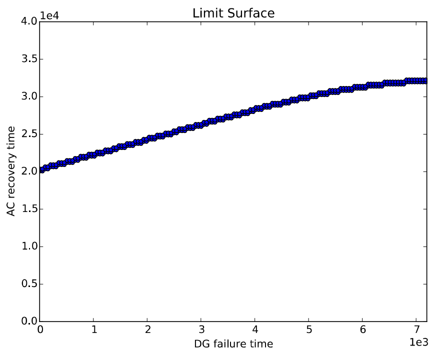
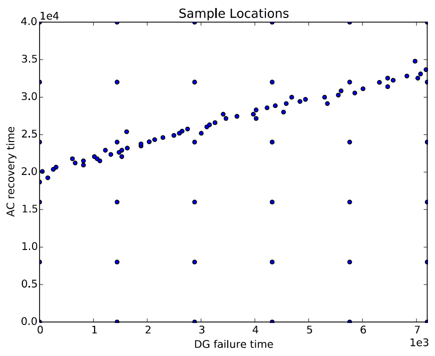
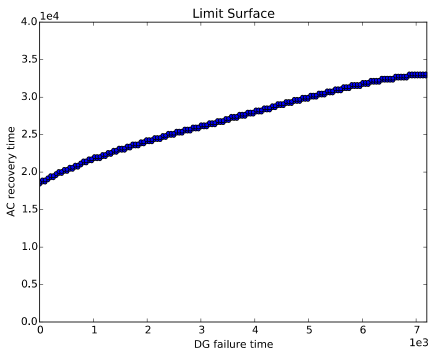
Iteration	Sample Locations	Estimated Limit Surface
1		
10		
30		
60		

Figure 22 – RELAP-7 limit surface: sample locations and the estimated limit surface for different adaptive sampling iterations (1, 10, 30 and 60)

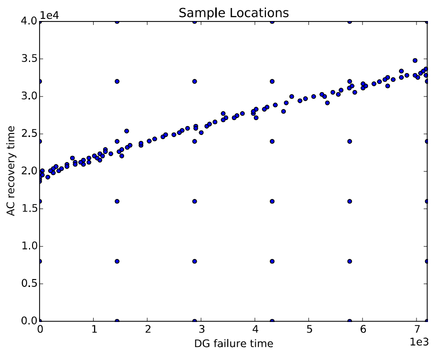
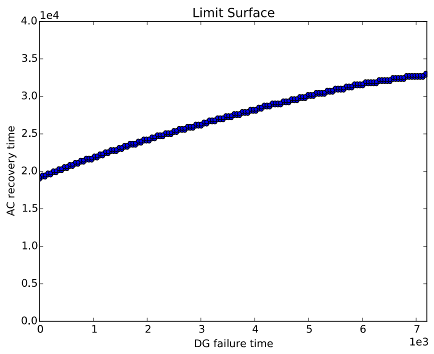
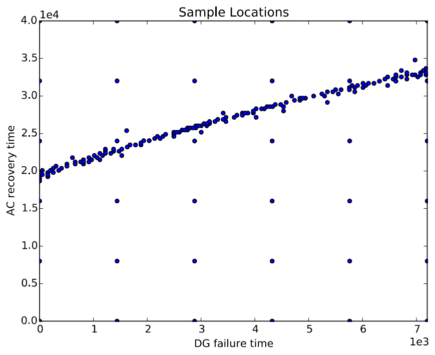
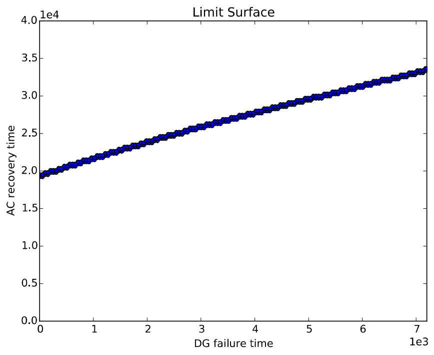
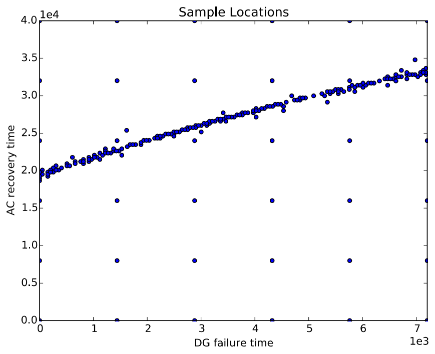
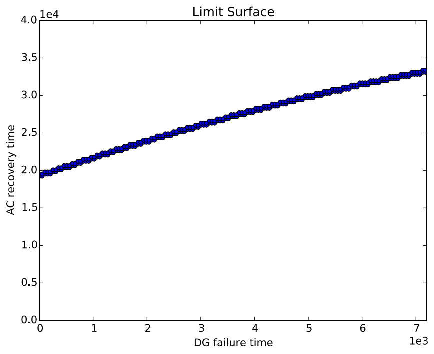
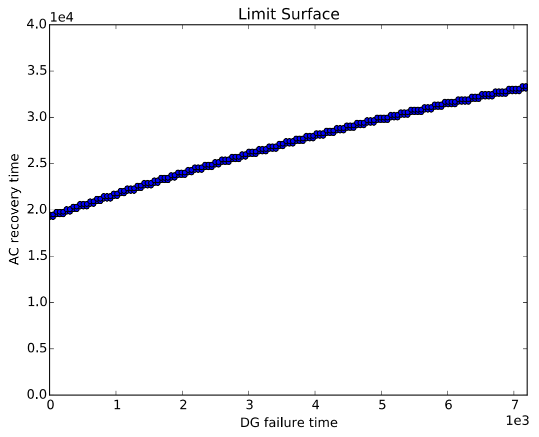
Iteration	Sample Locations	Estimated Limit Surface
100		
150		
185		

Figure 23 – RELAP-7 limit surface: sample locations and the estimated limit surface for different adaptive sampling iterations (100, 150 and 185)

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

We repeated the analysis of Section 6.2.2 for the case where the reactor power is set to 120% (i.e., a 20% power uprate). The scope of evaluating this new limit surface is to determine the reduction of the time to recovery for DGs. A 20% reactor power increase implies that clad temperature is increasing at a higher rate and thus the clad is reaching its melting temperature (2200 F) much faster.

We performed Steps 1 through 6 of Section 6.1 for the new data set and evaluated the new limit surface for the 120% test case and the results are shown in Figure 24.



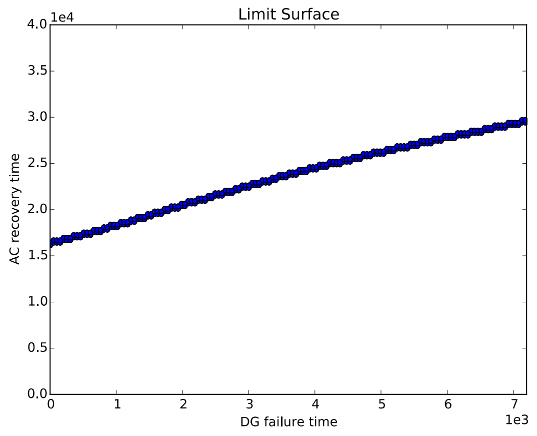


Figure 24 – Limit surface obtained for two different levels of core power: 100% (left) and 120% (right)

6.2 Uncertainty Quantification and Sensitivity Analysis

Another set of application of Surrogate Models relevant to the RISMC project is Uncertainty Quantification (UQ) and Sensitivity Analysis (SA). For these kinds of applications we are following a response-surface approach where a Surrogate Models is trained in the region of the input space of interest. This training process aims to reconstruct the system response in this limited region of the input space. Then, the forward propagation of the uncertainties of the input parameters is performed by using the surrogate models instead of the actual code. This process is summarized in Figure 25.

In Section 9 we show a comparison of several surrogate models for the forward propagation of uncertainties for a few test cases. In order to evaluate the performances of surrogate models we have decided to compare the following:

- First three moments of the figure of merits: mean, sigma and skewness
- Pearson coefficients of the input parameters
- Sensitivity coefficients of the input parameters

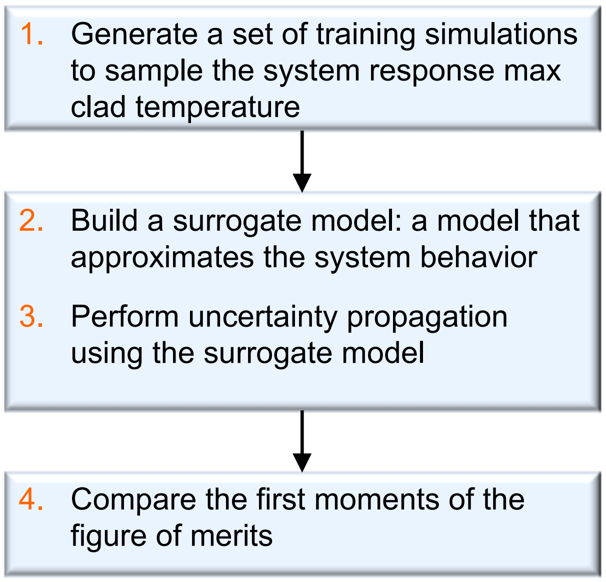

1. Generate a set of training simulations to sample the system response max clad temperature
2. Build a surrogate model: a model that approximates the system behavior
3. Perform uncertainty propagation using the surrogate model
4. Compare the first moments of the figure of merits

Figure 25 – Workflow used to perform UQ/SA using surrogate models

We followed the workflow shown in Figure 25 to perform UQ/SA on the PWR natural circulation loop shown in Section 5.2.1 by using SVM (see Section 8.2) as a surrogate Model. Figure 26 shows a plot of the response function for the PWR natural circulation loop using peak fuel temperature as the figure of merit.

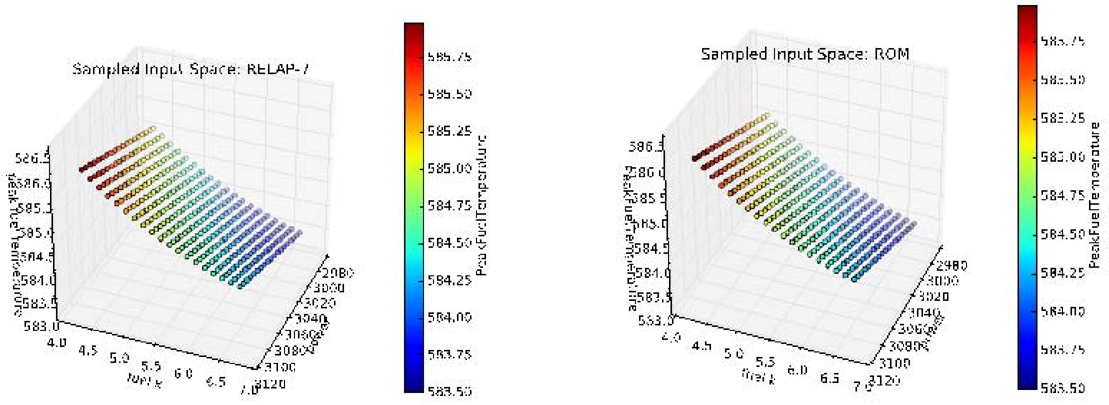
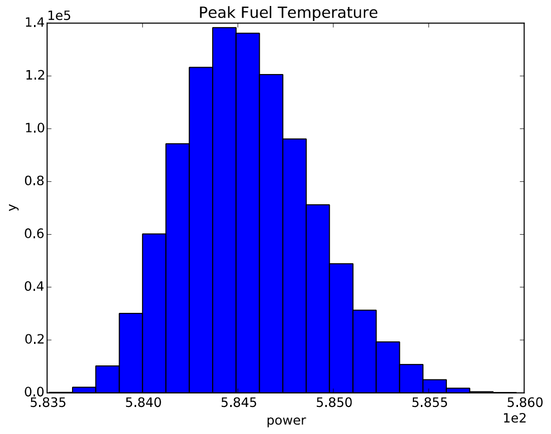


Figure 26 – Comparsion of the response function of peak fuel temperature: original RELAP-7 (left) and the one obtained by using the ROM (right)

Table 3 – Comparison of the first three statistical moments, sensitivity and Pearson coefficients of peak fuel temperature

	RELAP-7	ROM	Relative Error
mu	5.845 E+02	5.845 E+02	3.90 E-07
sigma	3.469 E-01	3.472 E-01	-9.64 E-04
skewness	3.446 E-01	3.434 E-01	3.72 E-03
K sensitivity	-6.664 E-01	-6.671 E-01	-1.15 E-03
P sensitivity	7.231 E-03	7.234 E-03	-3.65 E-04
K pearson	-9.056 E-01	-9.057 E-01	-1.90 E-04
P pearson	-3.972 E-04	-3.972 E-04	0.00 E+00



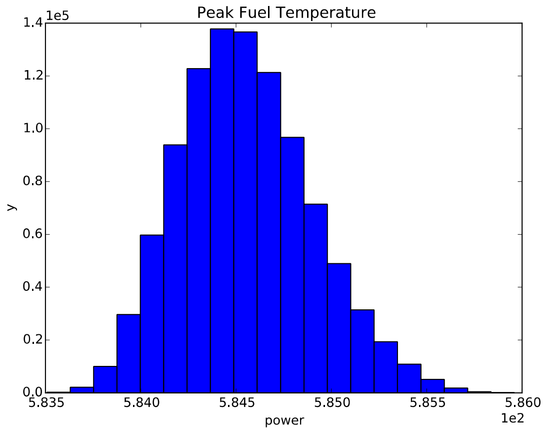


Figure 27 – Distribution of peak fuel temperature: RELAP-7 (left) and ROM (right)

6.3 ROM for Data Visualization

The need for software tools able to both analyze and visualize large amount of data generated by Dynamic PRA methodologies has been emerging only in recent years. In the past years, INL and the University of Utah have developed a software tool able to analyze multi-dimensional data: HDViz.

HDViz models the relations between output variables (e.g., maximum clad temperature) and stochastic/uncertain parameters as high-dimensional functions. In this respect, HDViz segments the domain of these high-dimension functions into regions of uniform gradient flow by decomposing the data based on its approximate Morse-Smale complex (see Figure 28). Points (i.e., simulation runs) belonging to a particular segmentation have similar geometric and topological properties, and from these it is possible to create compact statistical summaries of each segmentation. Such summaries are then presented to the user in an intuitive manner that highlights features of the dataset that are otherwise hidden.

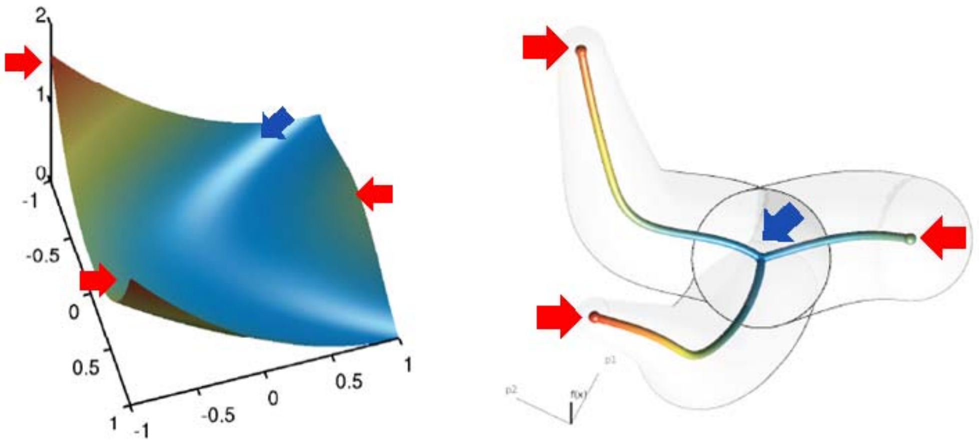


Figure 28 – Representation example of a 2-dimensional function in terms of crystals that connect local minima to local maxima. In this case, a single minima (blue arrow) and 3 maxima (red arrows) have been identified. Three crystal have also been determined; each one showing the path that connects a local minima to a local maxima

In [34] we presented a methodology that aims to visualize high dimensional data through topological segmentation of multidimensional surfaces. In our applications, such multi-dimensional spaces are determined by the set of n uncertain parameters $\{x_1, x_2, \ldots, x_n\}$ while safety related outcomes f, such as maximum core temperature, are considered for each simulation. Our topological tools aim to reconstruct the topological structure of $f(x_1, x_2, \ldots, x_n)$ (i.e., the response surface) in this n dimensional space:

$$\textit{system outcome} = f\left(\textit{uncertain parameters}\right)$$

We consider an alternative method for analyzing high dimensional data that models the simulation data as a scalar function f defined on a finite set of points X in $\mathbb{R}^n$ with scalar output, Y in $\mathbb{R}$. In this way, we can then summarize the structure of the data with respect to the topology imposed by a variable of interest that is classified as the output of our constructed function.

Various topological constructs and approximations exist that can be used to impose a structure on the arbitrary dimension, functional data, such as the contour tree [35], the reeb graph [36], and the Morse-Smale complex [37]. This paper makes use of the approximated Morse-Smale technique first established in [38] to present a hierarchical partitioning of the data based on estimated gradient flow behavior of the sampled data. In this way, we can still present the data in the same manner as the hierarchical clustering, but also take advantage of the scalar function property to offer alternative visualizations. A summary of this technique follows.

We partition the points in X based on their function values and gradient behavior using an approximated hierarchical Morse-Smale complex. That is, at the finest level of detail, points belong to the same cluster if they share gradient flow to the same local maximum and local minimum. Gradient flow is estimated by imposing a neighborhood graph on the data and using the steepest ascending/descending neighbor of a point to represent the gradient at each point sample. In this way, each point can be traced to a local maximum or minimum.

We can then merge clusters based on the persistence of their corresponding local extrema. We define persistence for an extremum as the minimum difference in function value between the extremum and its neighboring saddle points. We assume every saddle point is a simple saddle, thus every saddle will pair two local maxima or two local minima, and the merging of clusters is unambiguous as the adjusted gradient will be simulated to flow to the higher persistence extremum after two clusters are merged. An intuitive two-dimensional example is shown in Figure 29 where local maxima of various sizes are merged, in order, with a neighboring saddle point redirecting upward flow toward the maximum also sharing the same saddle point. The grey lines simulate this redirected flow and the circles represent the saddle locations. At each refinement, we simulate the new gradient flow with a simplified surface model in the figure.

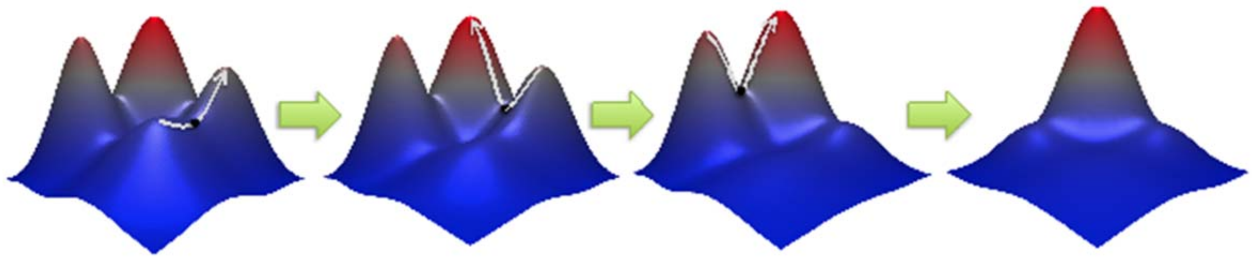


Figure 29 – A simulation of persistence simplification showing how the local maxima of this two dimensional function are hierarchically merged with a neighboring saddle redirecting upward flow to a neighboring maximum that shares the saddle point

We further the analysis by computing a summary curve via the following three step process: (1) perform inverse regression with each clusters data, (2) project the curves embedded in $\mathbb{R}^n$ to a two-dimensional viewing window using PCA [22], and (3) align the curves to meet at their shared extrema to maintain the coherency of the topological structure extracted. The resulting topological skeleton visualization encodes the projected average of each levelset of a cluster, a direction-independent estimate of the spread in domain space at each levelset presented as a transparent halo, and the sampling density at each levelset of a cluster encoded as the darkness of the halo. Figure 29 details each of these visualization components given a 2D example surface.

As a supplementary view, we project the data colored by its segment as well as the high-dimensional curve summary onto a set of stacked two-dimensional scatter plots. The horizontal axis is the output dimension which is aligned on all plots, and the separate vertical axes of each plot are the individual input dimensions. The curve summary gives the average location at each levelset and each curve now uses a dimension-specific standard deviation for the width of the transparent halo. Furthermore, the color of the halo is held constant as the sampling density is better encoded by the overlaying of the data points themselves. In this way, we can begin to distinguish with respect to each individual dimension how the segments differ. Figure 30 shows a labeled version of this interface. The topological skeleton makes it more clear at a glance that there are four distinct segments, yet gives no underlying information about the sampled data in each cluster. This is why we believe that the combination of these techniques represents the best view of the data. More details of the visualization pipeline as well as additional views using the same input data format are described in prior works [38,39].

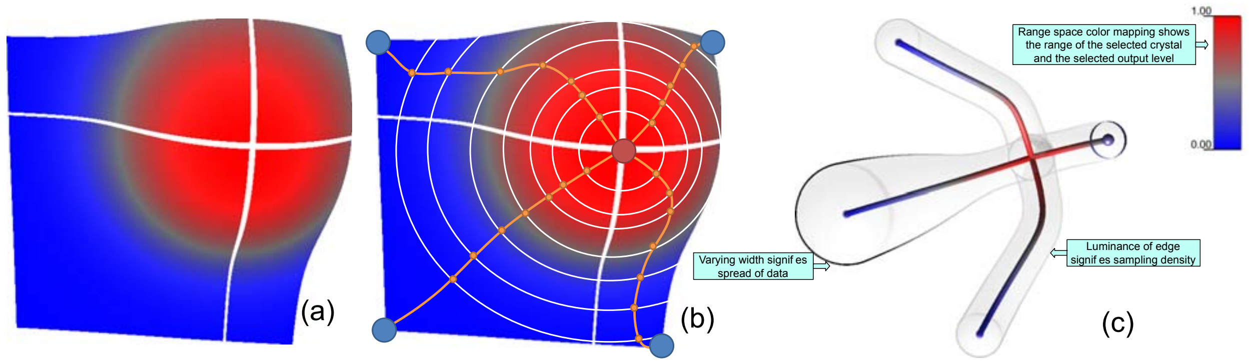

(a)
(b)
(c)
Range space color mapping shows the range of the selected crystal and the selected output level
1.00
0.00
Varying width signif es spread of data
Luminance of edge signif es sampling density

Figure 30 – An abstract view of our visualization technique on a 2D smooth surface: The surface is first segmented into areas of uniform gradient flow (a), then each levelset (modeled by a white line) is averaged to a single point and consecutive levelsets are connected to form a curve per segment (orange curves), and finally our result is shown with labels for each visualization cue

We apply clustering algorithms [40,41] based on the Morse-Smale complex [37,42] on the dataset obtained from RAVEN for the BWR SBO analysis [43]. In essence, we aim to reconstruct the response surface (i.e., max clad temperature) topological structure in a *d*-dimensional space where *d* is the number of uncertain parameters.

We further obtain a topological summary for each cluster and try to infer the correlations between simulation parameters and system observations. The objective is to find the combination of conditions (in the form of input simulation parameters) that can cause core damage. Before analysing the data we performed a series of pre-processing procedures:

- Data standardization: The above data is pre-processed with a standardization process. Since different parameters may be measured on different scales and the range of values differ from each dimension, some parameters may dominate the results of the analysis. We employ a z-score data standardization process so that all dimensions are on the same scale. For values of each dimension, we subtract the mean and divide by the standard deviation.

- Dimension reduction: Upon further observations of the nature of the simulation, we further transform the data by reducing the number of dimensions. In particular, we introduce 3 new dimensions by combining 3 pairs of dimensions from the raw dataset:

    - ACPowerRecoveryTime: min {RecoveryTimeDG; OFFsitePowerRecoveryTime}.

    - SRVstuckopen: min {SRV1stuckopen; SRV2stuckopen}.

    - CoolingFailtoRunTime: max {HPCIFailToRunTime; RCICFailToRunTime}.

The 9D case includes then the following input variables:

1. FailureTimeDG

2. ACPowerRecoveryTime

3. SRVstuckOpenTime

4. cladFailureTemperature

5. CoolingFailtoRunTime

6. Reactor power

7. ADSactivation-TimeDelay

8. firewaterTime

9. TotalBatteryLife

The output variable is the maxCladTemp (MT).

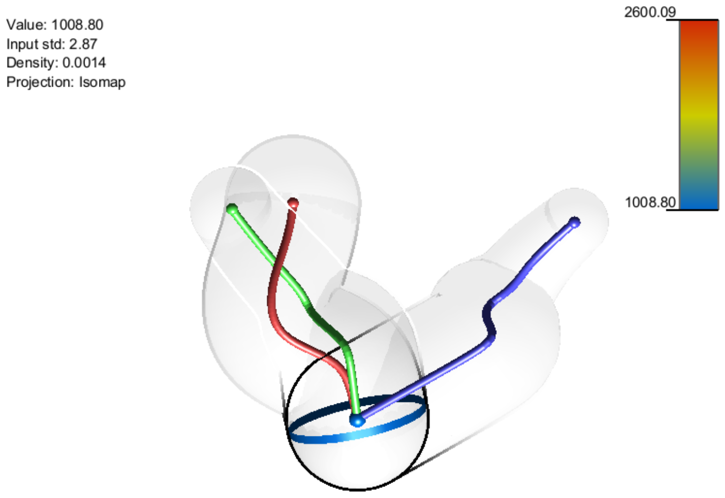


Figure 31 – Topological summary

Using HDViz, from 9D-MT-all-3C, we were able to obtain 3 clusters as shown in Figure 32. The topological structure of the clad max temperature as a 9-dimensional surface was characterized by a single local minimum and 3 local maxima as indicated in Table 4.

Figure 32 shows the projection of the three crystals for each dimension including their regression curves: x-axis corresponds to output variable (maximum clad temperature) while y-axis corresponds to input variable. From Figure 32, by looking at the regression curve obtained, we can see that a high value of clad temperature is reached, for all 3 crystals, for a late AC recovery time. As expected a late AC recovery time is a necessary condition to reach core damage. The same conclusions can be drawn for FW injection time, a late FW injection time guarantees core damage as well.

Failure time of DGs differentiates the three crystals, i.e., a late DG failure time is not sufficient to guarantee system success. In fact, by looking at the green crystal regression curve, a late failure time of DGs coupled with an early SRV stuck-open event and an early failure of the high-pressure injection system (both RCIC and HPCI) leads to core damage. By looking at the regression curve of the purple crystal, core damage condition was reached for an early DGs failure time and an early failure of the high-pressure injection system (both RCIC and HPCI).

Table 4 – Minima and maxima of the crystals of Figure 10

Crystal color (see Figure 10)	*Min*	*Max*
Red	1008.80	2600.09
Green	1008.80	2597.20
Blue	1008.80	2534.16

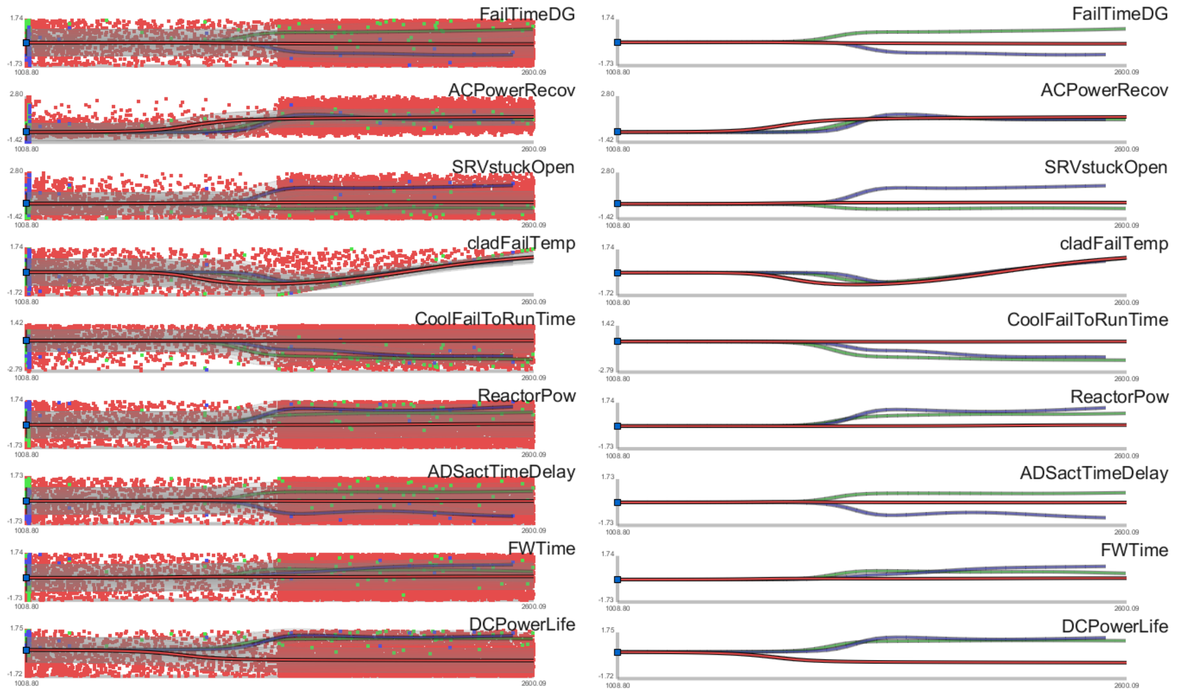


Figure 32 – Inverse coordinate plots with (left) and without (right) points projection

7. TEMPORAL SURROGATE MODEL

In the previous section we introduced the concept of response surface methods and surrogate models as tools to predict an approximated $\boldsymbol{\Theta}(\boldsymbol{s})$ (which represents, for example, a simulated system response under an accident scenario) for a set of conditions specified in $\boldsymbol{s}$. The vector $\boldsymbol{s}$ contains elements s_d such as timing and sequencing of events (e.g., recovery time of AC power, failure time of core cooling injection). Note that the value $\boldsymbol{\Theta}(\boldsymbol{s})$ is a scalar and, thus, does not contain any temporal evolution type of information.

We extend the concept of ROM in order to be able to handle time dependent $\boldsymbol{\Theta}(\boldsymbol{s})$: given $\boldsymbol{s}$, $\boldsymbol{\Theta}(\boldsymbol{s},t)$ is a time dependent variable. In this case, the training consists of N points:

$$(\boldsymbol{s}_i, \boldsymbol{\mathcal{H}}(\boldsymbol{s},t)_i) \;\; i = 1, \dots, N \qquad \textbf{Eq. 8}$$

Our approach is to start by dividing the temporal scale into intervals (assumed here to be of equal length but it is not required):

$$t = [t_1, \dots, t_T] \qquad \textbf{Eq. 9}$$

For each time point t_k $(k = 1, \dots, T)$ we consider the subset of points:

$$(\boldsymbol{s}_i, \boldsymbol{\mathcal{H}}(\boldsymbol{s},t_k)_i) \;\; i = 1, \dots, N \qquad \textbf{Eq. 10}$$

and we build the corresponding $\boldsymbol{\Theta}(\boldsymbol{s})_k$. Thus, now we have a set of ROMs for each time point t_k $(k = 1, \dots, T)$. The temporal predictor $\Psi(\boldsymbol{x},t)$ is simply the vector of:

$$\boldsymbol{\Psi}(\boldsymbol{x},t) = [\boldsymbol{\Theta}(\boldsymbol{s})_1, \dots, \boldsymbol{\Theta}(\boldsymbol{s})_k, \dots, \boldsymbol{\Theta}(\boldsymbol{s})_T] \qquad \textbf{Eq. 11}$$

In our applications, when each of the data points has been generated by safety analysis codes (e.g., RELAP):

- $\boldsymbol{s}$ is the configuration of the simulation (e.g., timing of events, values associated with uncertain parameters)
- $\boldsymbol{\Theta}(\boldsymbol{s},t)$ is the simulation associated with $\boldsymbol{s}$.

We performed a few tests with different types of datasets in order to identify performances and limitations of this algorithm. Figure 33 (left) shows a set of $n = 20$ simulations, i.e. $\boldsymbol{\mathcal{H}}(\boldsymbol{s},t)_i$ ($i = 1, \dots, 20$), generated by sampling two stochastic parameters, i.e. $\boldsymbol{s}_i = [s_1, s_2]$.

We initially divided the time scale uniformly [0,2500] into $T = 100$ intervals and for each time point t_k $(k = 1, \dots, 100)$ we considered the data points $(\boldsymbol{x}_i, \boldsymbol{\mathcal{H}}(\boldsymbol{s},t_k)_i)$ $(i = 1, \dots, 20)$ and built the reduced order models $\boldsymbol{\Theta}(\boldsymbol{s})_k$.

We then tested the temporal predictor:

$$\boldsymbol{\Psi}(\boldsymbol{s},t) = [\boldsymbol{\Theta}(\boldsymbol{s})_1, \dots, \boldsymbol{\Theta}(\boldsymbol{s})_{100}] \qquad \textbf{Eq. 12}$$

for several $\boldsymbol{s}_j$ $(j \neq i)$ and compared them with the simulated $\boldsymbol{\Theta}(\boldsymbol{s},t)$.

Figure 33 (right) shows the predicted scenario $\Psi(\boldsymbol{s},t)$ (green line) and the actual simulated scenario $\boldsymbol{\mathcal{H}}(\boldsymbol{s},t)$. For this particular case we built $\Psi(\boldsymbol{s},t)$ using Gaussian Process Models as a basic ROM. A useful feature is that these algorithms are also capable of providing the uncertainty associated with the predicted results.

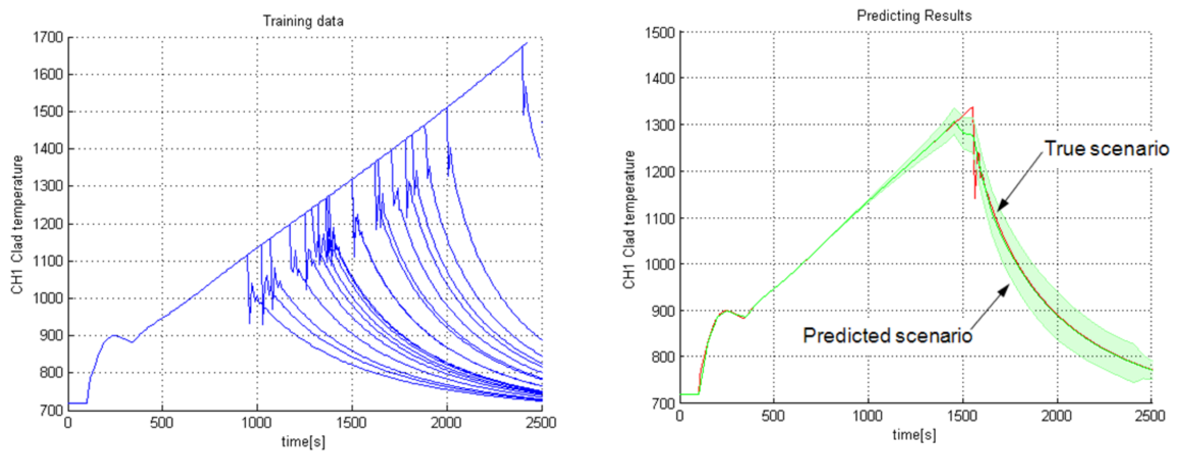


Figure 33 – Initial protoype results of temporal surrogate model

7.1 RAVEN Implementation

In order to allow the RAVEN statistical framework to create temporal surrogate models, we started in this fiscal year to develop the needed basic software infrastructure. In the following fiscal year plan to extend such capabilities. This development was accomplished by modifying the ROM class to:

- Accept a set of multivariate histories as training data
- Instantiate and train a matrix of surrogate models
- Create a multivariate history as predicted data

The only requirement needed is that the set of training histories contains the identical set of input/output variables.

We slightly modified the algorithms shown in Eq. 11 in order to give the flexibility to the algorithm to accept histories that have different time length. Depending on the final outcome or depending to other conditions the temporal length of each simulation run may change considerably.

The algorithm performs the following:

- Each history is sampled in time using the same number of sampling points. Two sampling strategies are available (see Figure 34):
    - Uniform: the temporal distance between two samples is constant

  - Derivative: the sample locations are concentrated in regions with higher temporal derivatives. These can be based on both the first or second derivative.

- The temporal predictor is built according to Eq. 11 but with the assumption that each element $\mathbf{\Theta}(\boldsymbol{s})_t$ represents the surrogate model for the sample t

- Since the time location of each sample t changes from history to history an additional vector of surrogate models $\mathbf{\Upsilon}$ is created:

$$\mathbf{\Upsilon} = [\mathbf{T}(\boldsymbol{s})_1, \dots, \mathbf{T}(\boldsymbol{s})_k, \dots, \mathbf{T}(\boldsymbol{s})_T] \qquad \textbf{Eq. 13}$$

  where each element $\mathbf{T}(\boldsymbol{s})_k$ is a surrogate model that predicts the time location of each sample k $(k = 1, \dots, T)$

- Full prediction is performed by solving the system of surrogate models

$$\begin{cases} \mathbf{\Psi}(\boldsymbol{x}, t) = [\mathbf{\Theta}(\boldsymbol{s})_1, \dots, \mathbf{\Theta}(\boldsymbol{s})_k, \dots, \mathbf{\Theta}(\boldsymbol{s})_T] \\ \mathbf{\Upsilon} = [\mathbf{T}(\boldsymbol{s})_1, \dots, \mathbf{T}(\boldsymbol{s})_k, \dots, \mathbf{T}(\boldsymbol{s})_T] \end{cases} \qquad \textbf{Eq. 14}$$

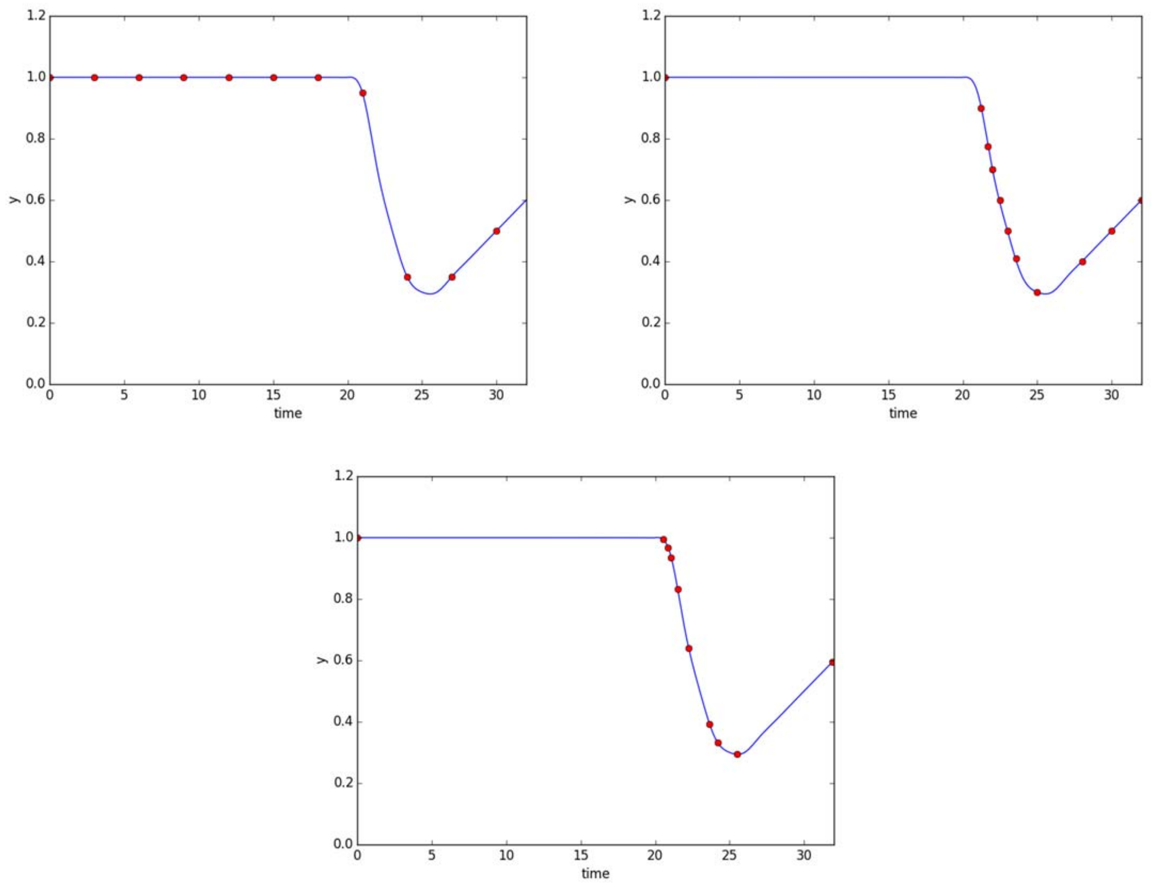


Figure 34 – Example of time series sampling steategies: uniform (top left), first-derivative based (top right) and second-derivative based (bottom).

We performed an initial testing of the developed code on a simple test case: a heating-cooling lumped system. This system behaves as follows (see Figure 35-left):

- The initial condition is fixed: temperature fixed at 800 Fahrenheit. The cooling system slowly decreases the system temperature
- At time $t = tSBO$, the cooling stops and the system heats up, i.e., the system temperature increases
- At time $t = tREC$, the cooling restarts and the system temperature decreases

Two uncertain parameters have been chosen: $tSBO$ and $tREC$. A training set has been created on a 6×6 grid in the 2-dimensional input space (see Figure 35-right). We chose to create a temporal surrogate model based on Gaussian Process models by using 10 uniform time discretizations (i.e., $k = 1, \dots, 10$). As can be seen from Figure 36, the predicted and the "true" transient match pretty well.

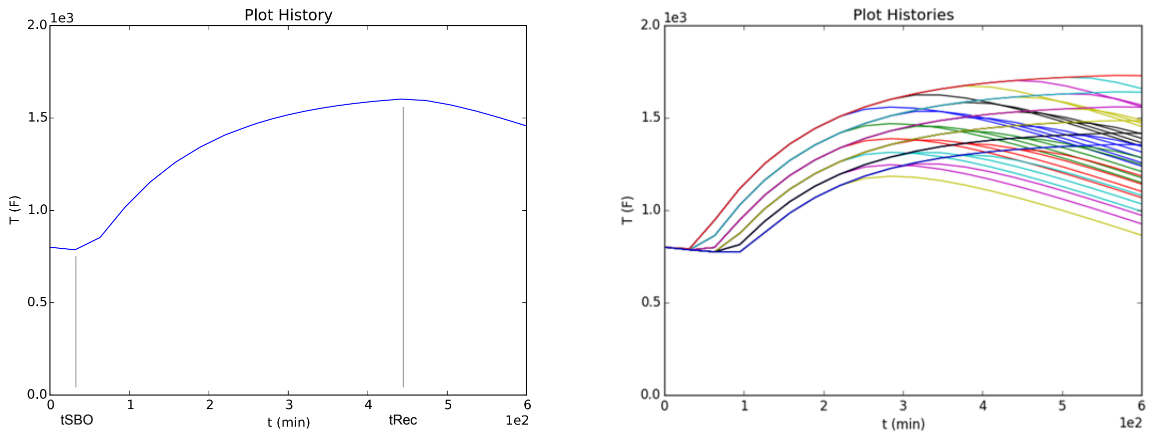
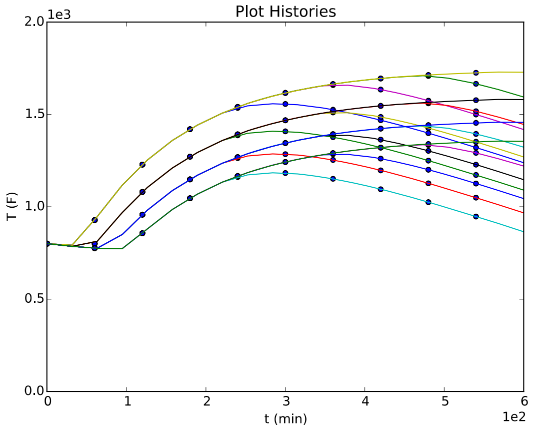


Figure 35 – Example of transient as function of two parameters, tREC and tSBO, (left) and plot of the training data (right)



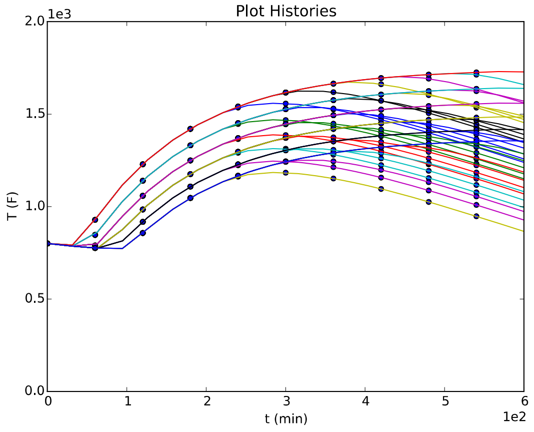


Figure 36 – Comparison of predicted transients generated by the temporal surrogate model (dots) and the true transients (line) for two different sampling strategies (4 × 4 and 6 × 6 sampling grid)

8. SURROGATE MODELS IN RAVEN

In this section we describe a subset of surrogate models that are available within RAVEN. We chose a limited set of surrogate models that can be of interest for RISMC type of analyses.

8.1 KNN Classifier and KNR Regressor

The *K* Nearest Neighbor algorithm [25] 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.

8.2 Support Vector Machines

Given a set of *N* multi-dimensional samples $\boldsymbol{x}_i$ and their associated results $y_i = \pm 1$ (e.g., $y_i = +1$ for system success and $y_i = -1$ for system failure), the SVM algorithm [44,45] finds the boundary (i.e., the decision function) that separates the set of points having different y_i. The decision function lies between the support hyper-planes, which are required to:

- Pass through at least one sample of each class (called support vectors)

- Not contain samples within them

For the linear case, see Figure 37, the decision function is chosen such that distance between the support hyper-planes is maximized.

From a mathematical point of view, the hyper plane can be written in this form:

$$\boldsymbol{w}\,\boldsymbol{x} - b = 0 \qquad \text{Eq. 15}$$

Given this formulation, the SVM algorithm aims to:

$$\begin{gathered} minimize \;\; \|\boldsymbol{w}\| \\ subject\ to\ y_i(\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle - b) \geq 1 \end{gathered} \qquad \text{Eq. 16}$$

Without going into the mathematical details, the determination of the hyper-planes is performed recursively and updated every time a new sample has been generated. Figure 37 shows the SVM decision function and the hyper-planes for a set of points in a 2-dimensional space having two different outcomes: $y_i = +1$ (green) and $y_i = -1$ (red).

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

$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \exp\left(-\frac{\|\boldsymbol{x_i} - \boldsymbol{x_j}\|}{2\sigma^2}\right) \qquad \text{Eq. 17}$$

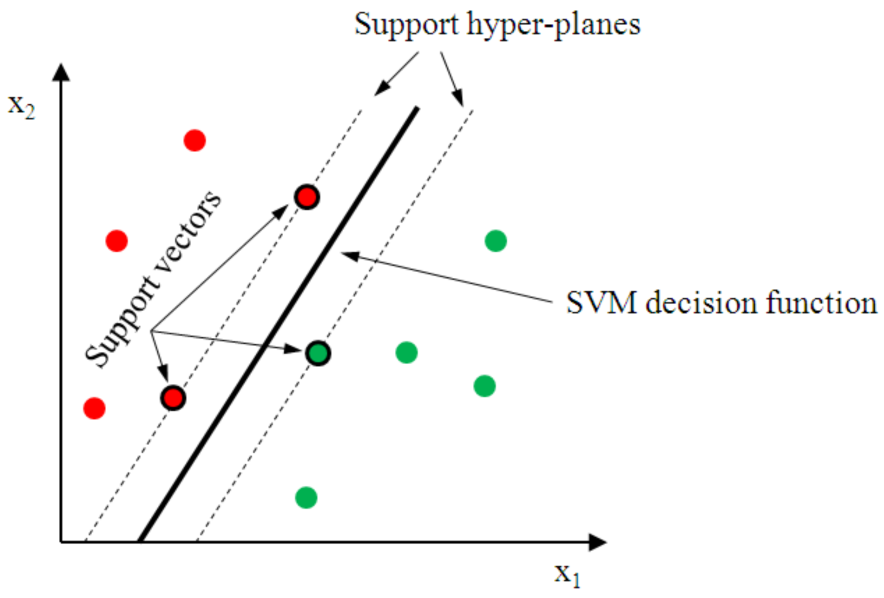


Figure 37 – Limit surface evaluation using SVMs

8.3 Gaussian Process Models

Gaussian processes (GPs) [21] are algorithms that extend multivariate Gaussian distributions to infinite dimensionality. A Gaussian process generates a dataset 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, $\boldsymbol{y} = \{y_1, \dots, y_n\}$, 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:

$$k(x, x') = \sigma_f^2 \, exp\left[\frac{-(x - x')^2}{2\, l^2}\right] \qquad \text{Eq. 18}$$

where the maximum allowable covariance is defined as σ_f^2 — this should be high for functions which cover a broad range on the y axis. If $x \approx x'$, 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 $k(x, x') \approx 0$, 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:

$$y = f(x) + \mathcal{N}(0, \sigma_n^2) \qquad \text{Eq. 19}$$

The new kernel function can be written as:

$$k(x, x') = \sigma_f^2 \, exp\left[\frac{-(x - x')^2}{2\, l^2}\right] + \sigma_n^2 \delta(x, x') \qquad \text{Eq. 20}$$

So given n observations $\boldsymbol{y}$, 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:

$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix} \qquad \text{Eq. 21}$$

$$K_* = [k(x_*, x_1) \quad \cdots \quad k(x_*, x_n)] \qquad \text{Eq. 22}$$

$$K_{**} = k(x_*, x_*) \qquad \text{Eq. 23}$$

2. The basic assumption of GPM is that

$$\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \sim \mathcal{N}(0, \begin{bmatrix} K & K_*^{\,T} \\ K_* & K_{**} \end{bmatrix}) \qquad \text{Eq. 24}$$

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

$$\bar{y}_* = K_* \, K^{-1} \, \boldsymbol{y} \qquad \text{Eq. 25}$$

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

$$var(y_*) = K_{**} - K_* \, K^{-1} \, {K_*}^T \qquad \text{Eq. 26}$$

8.4 Generalized Polynomial Chaos

In general, Polynomial Chaos Expansion (PCE) methods seek to represent the simulation code as a combination of polynomials of varying degree and combination in each dimension of the input space. Originally Wiener proposed expanding in Hermite polynomials for Gaussian-normal distributed variables. Askey and Wilson generalized this method 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.

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). 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; the other uses the Wiener-Askey polynomials.

We consider a simulation code that produces a quantity of interest u as a function $u(Y)$ whose arguments are the uncertain, distributed input parameters $Y = [Y_1, \dots, Y_n, \dots, Y_N]$. A particular realization ω of Y is expressed by $Y(\omega)$, and a single realization of the entire input space results in a solution to the function as $u(Y(\omega))$. We acknowledge obtaining a realization of $u(Y)$ may take considerable computation time and effort, and may be solved nonlinearly and without analytic solution. There may be other input parameters that contribute to the solution of $u(Y)$; we neglect these, as our interest is Y in the uncertainty space. In addition, it is possible that the quantity of interest $u(Y)$is an integrated quantity or some norm of a value that is temporally or spatially distributed; in any case, we restrict $u(Y(\omega)$ to a single scalar output.

We expand $u(Y)$ in orthonormal multidimensional polynomials $\Phi_k(Y)$, where k is a multi-index tracking the polynomial in each axis of the polynomial Hilbert space, and $\Phi_k(Y)$ is constructed as

$$\Phi_k(Y) = \prod_{n=1}^{N} \Phi_{k_n}(Y_n) \qquad \text{Eq. 27}$$

where $\Phi_{k_n}(Y_n)$ is a single-dimension Wiener-Askey orthonormal polynomial of order k_n and $k = [k_1, \dots, k_n, \dots, k_N]$. The gPC for $u(Y)$ using this notation is

$$u(Y) \approx \sum_{k \in \Lambda(L)} u_k \, \Phi_k(Y) \qquad \text{Eq. 28}$$

where u_k is a weighting polynomial coeffcient. The polynomials used in the expansion are determined by the set of multi-indices $\Lambda(L)$, where L is a truncation order for isotropic methods. In the limit that Λ contains all possible combinations of polynomials of any order, Eq. 28 is exact. Practically, however, Λ is truncated to some finite set of combinations.

Using the orthonormal properties of the Wiener-Askey polynomials,

$$\int_{\Omega} \Phi_k(Y)\, \Phi_{\hat{k}}(Y)\, \rho(Y)\, dY = \delta_{k\hat{k}} \qquad \text{Eq. 29}$$

where $\rho(Y)$ is the combined PDF of Y, is the multidimensional domain of Y, and is the Dirac delta, we can isolate an expression of the polynomial expansion coefficients.

We multiply both sides of Eq. 28 by $\Phi_{\hat{k}}(Y)$, integrate both sides over the probability weighted input domain, and sum over all $\hat{k}$ to obtain the coeffcients, sometimes referred to as polynomial expansion moments,

$$u_k = \frac{\langle u(Y)\, \Phi_k(Y) \rangle}{\langle \Phi_k(Y)^2 \rangle} = \langle u(Y)\, \Phi_k(Y) \rangle \qquad \text{Eq. 30}$$

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

$$\langle f(Y) \rangle = \int_{\Omega} f(Y)\, \rho(Y)\, dY \qquad \text{Eq. 31}$$

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, we can harness the properties of Gaussian quadratures because of the probability weights and domain.

8.5 Support Vector Regressor

A version of SVM for regression called support vector regression (SVR) was proposed in [47]. The model produced by support vector classification (as described above) depends only on a subset of the training data, because the cost function for building the model does not care about training points that lie beyond the margin. Analogously, the model produced by SVR depends only on a subset of the training data, because the cost function for building the model ignores any training data close to the model prediction.

Training the original SVR means solving:

$$\begin{aligned} &\textit{minimize } \frac{1}{2}\|\boldsymbol{w}^2\| \\ &\textit{subject to } \begin{cases} y_i - \langle \boldsymbol{w}, \boldsymbol{x_i} \rangle - b \leq \varepsilon \\ \langle \boldsymbol{w}, \boldsymbol{x_i} \rangle + b - y_i \leq \varepsilon \end{cases} \end{aligned} \quad \text{Eq. 32}$$

where $\boldsymbol{x_i}$ is a training sample with target value y_i. The inner product $\langle \boldsymbol{w}, \boldsymbol{x_i} \rangle + b$ is the prediction for that sample, and ε is a free parameter that serves as a threshold: all predictions have to be within an ε range of the true predictions.

8.6 Sheppard Method

The Inverse-Weight interpolator [48], also known as Sheppard interpolator, is ROM completely different from SVM. It performs predictions not on binary but on continuous outcomes. The starting point is a set of N data points $(\boldsymbol{x}_i, u_i)$ $i = 1, \ldots, N$ where $\boldsymbol{x}_i$ are the coordinate in the input space $D \subset \mathbb{R}^n$ and u_i is the outcome. The Inverse-Weight interpolator can be represented as a function $u(\boldsymbol{x})$ that, given a coordinate in the input space $\boldsymbol{x}$, generates a prediction on u:

$$u(\boldsymbol{x}): \boldsymbol{x} \to \mathbb{R}, \quad \boldsymbol{x} \in D \subset \mathbb{R}^n \quad \text{Eq. 33}$$

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

$$u(\boldsymbol{x}) = \begin{cases} \dfrac{\sum_{i=1}^{N} w_i(\boldsymbol{x})\, u_i}{\sum_{i=1}^{N} w_i(\boldsymbol{x})} & \text{if } d(\boldsymbol{x}, \boldsymbol{x}_i) \neq 0 \\ u_i & \text{if } d(\boldsymbol{x}, \boldsymbol{x}_i) = 0 \end{cases} \quad \text{Eq. 34}$$

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

$$w_i(\boldsymbol{x}) = \left(\frac{1}{d(\boldsymbol{x}, \boldsymbol{x}_i)}\right)^p \quad \text{Eq. 35}$$

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

9. COMPARISON AND CONVERGENCE STUDIES OF CLASSIFIERS

This section focus in detail on some of the classifiers that are available within RAVEN. In particular we focus on SVM (see Section 9.1) and KNN (see Section 9.2). For both algorithms we tested their adaptive sampling performances using three different test cases: Rosenbrock function, triangular function and the RELAP-7 TMI case. In addition we performed a series of convergence studies for both SVM and KNN. A summary of the structure of this section is shown below:

Classifier	Test Cases			Convergence study
	Rosenbrock	Triangular	RELAP-7 TMI	
SVM	Section 9.1.1	Section 9.1.2	Section 9.1.3	Section 9.1.4
KNN	Section 9.2.1	Section 9.2.2	Section 9.2.3	Section 9.2.4

9.1 SVM Classifier

The first classifier to be tested is the SVM with Gaussian Kernel (i.e., radial basis function - rbf). In the RAVEN input file such surrogate model is specified as below:

```
<ROM name="AccelerationROM" subType="SciKitLearn">
  <Features>x1,x2</Features>
  <Target>goalFunction</Target>
  <SKLtype>svm|SVC</SKLtype>
  <kernel>rbf</kernel>
  <gamma>0.1</gamma>
  <C>10.0</C>
</ROM>
```

RAVEN Input block 1 – SVM definition for the Rosenbrock test

9.1.1 Rosenbrock Test

The Rosenbrock test was performed by setting the following information in the RAVEN input file:

```
<Convergence limit = "3000" weight = "value" persistence = "0">1.0e-
4</Convergence>
```

RAVEN Input block 2 – Convergence settings for the Rosenbrock test: SVM case

An initial grid 6×6 was used as training set. The limit surface was obtained after 645 samples and the obtained expected value was 2.42 E-01 (theoretical values equal to 2.43 E-1). Figure 38 and Figure 39 shows the evolution of the estimate of the limit surface and the locations of the samples throughout the adaptive sampling process.

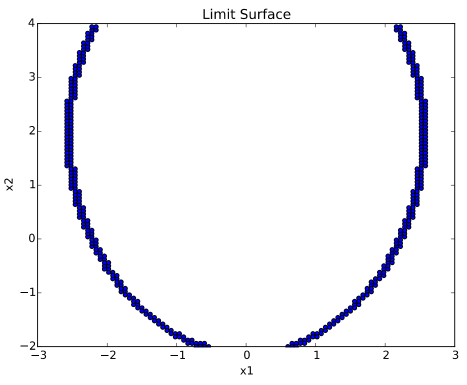
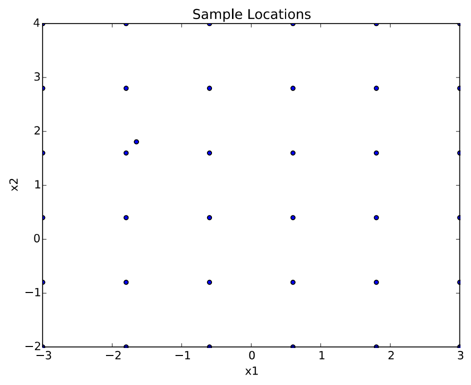
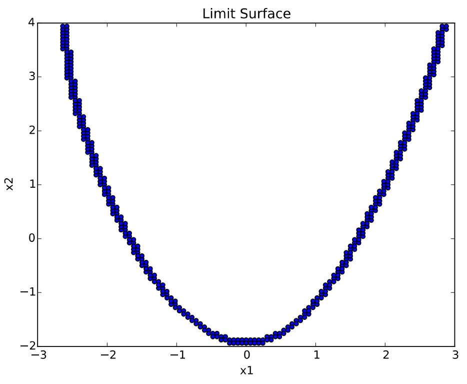
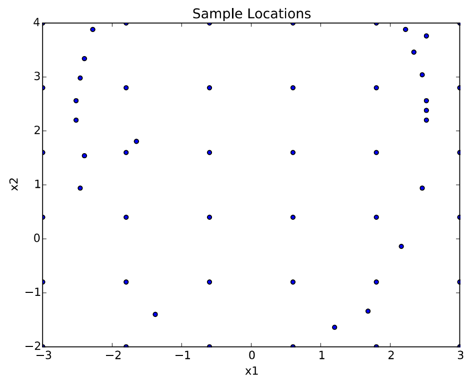
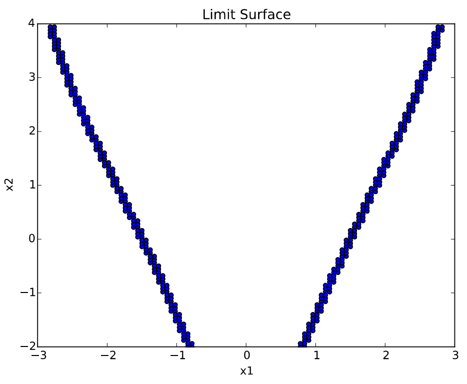
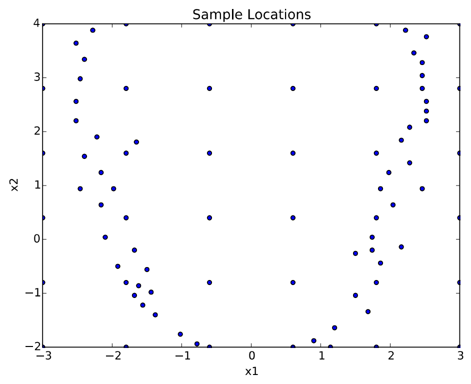
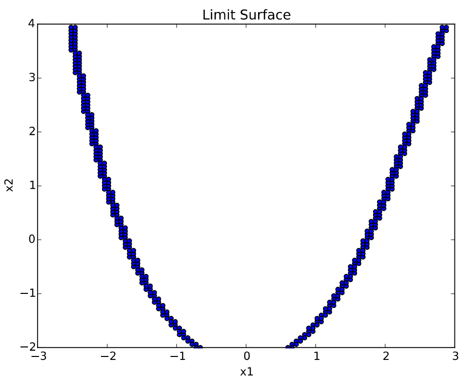
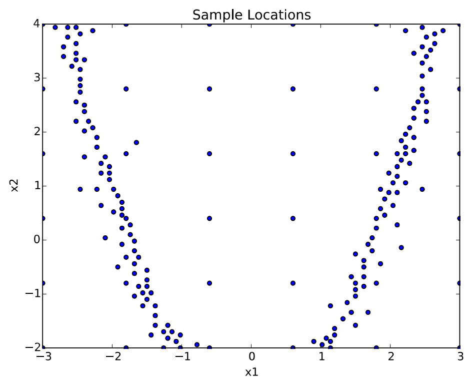
Iteration	Estimated limit surface	Sample locations
1		
20		
50		
150		

Figure 38 – SVM: calculation evolution of the Rosenbrock limit surface

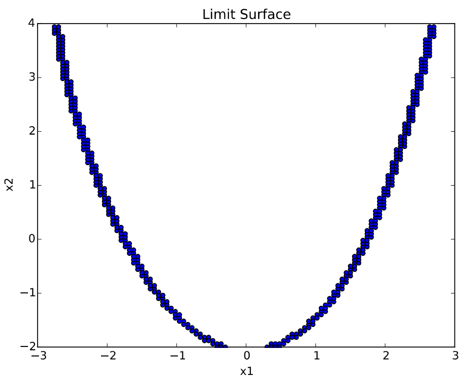
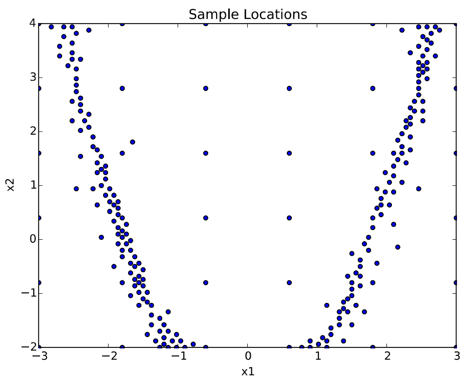
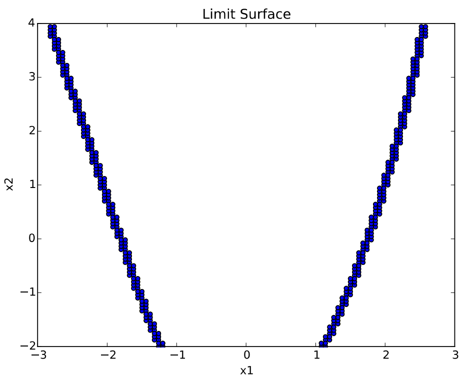
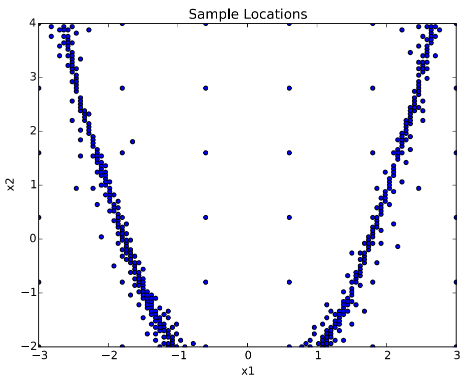
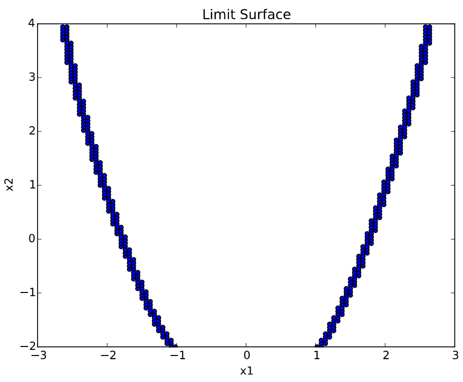
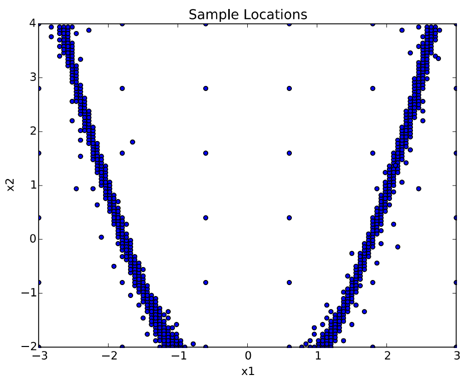
Iteration	Estimated limit surface	Sample locations
200		
400		
645		

Figure 39 – SVM: calculation evolution of the Rosenbrock limit surface

9.1.2 Triangular Test

We tested again SVM using Gaussian Kernel as indicated below.

```
<ROM name="AccelerationROM" subType="SciKitLearn">
  <gamma>50.0</gamma>
  <Features>x1,x2</Features>
  <Target>goalFunction</Target>
  <SKLtype>svm|SVC</SKLtype>
  <kernel>rbf</kernel>
  <C>1.0</C>
</ROM>
```

RAVEN Input block 3 – SVM definition for the triangular test

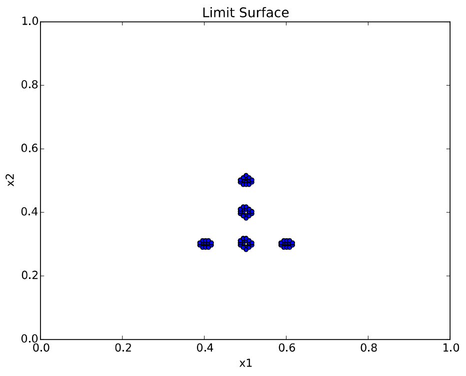
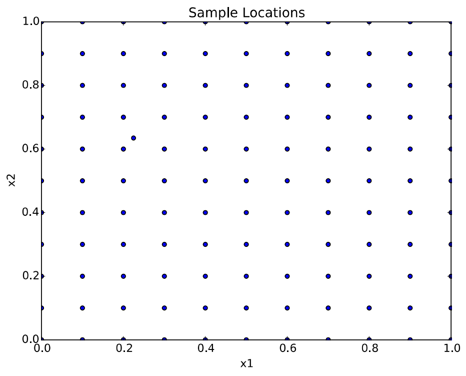
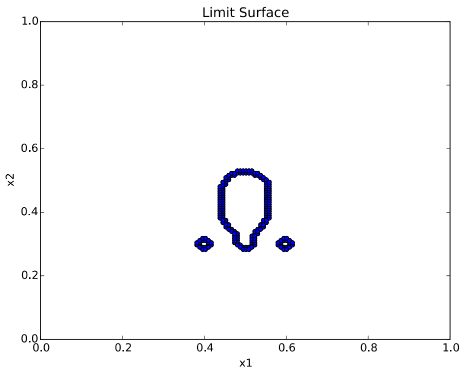
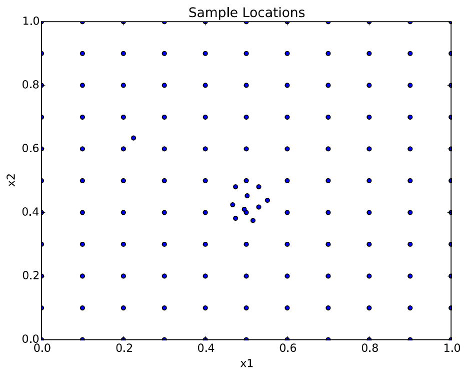
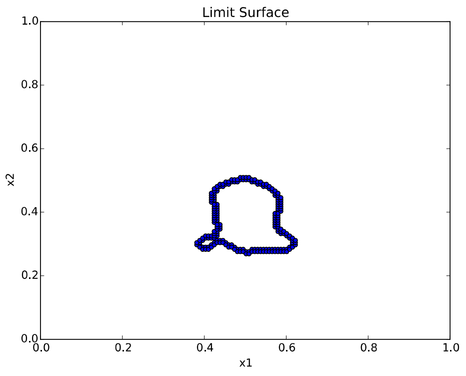
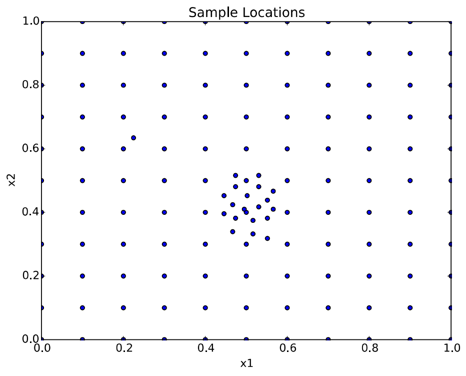
Iteration	Estimated limit surface	Sample locations
1		
10		
20		

Figure 40 – SVM: calculation evolution of the triangular limit surface

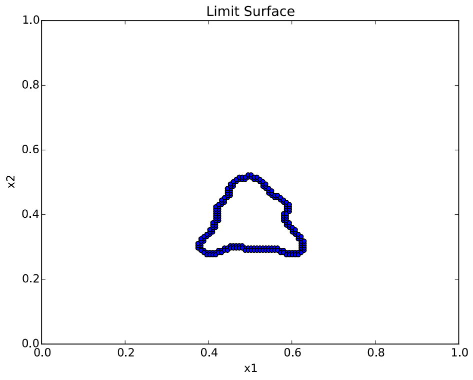
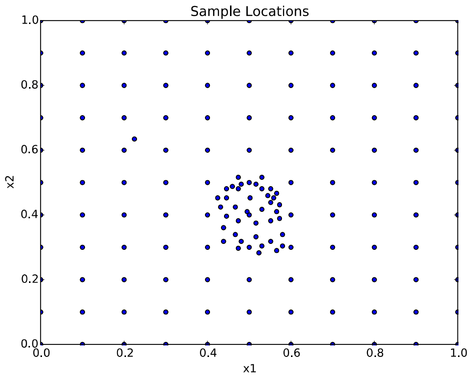
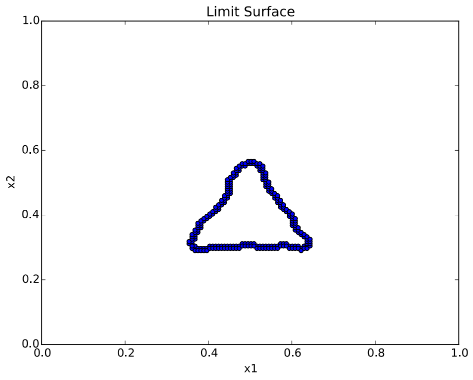
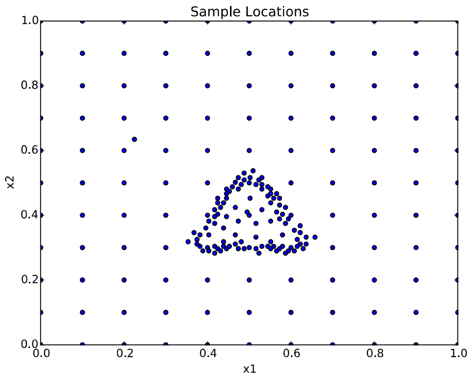
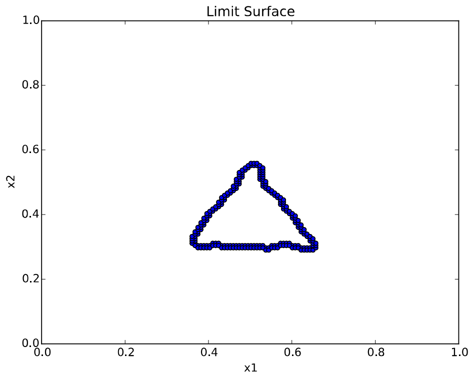
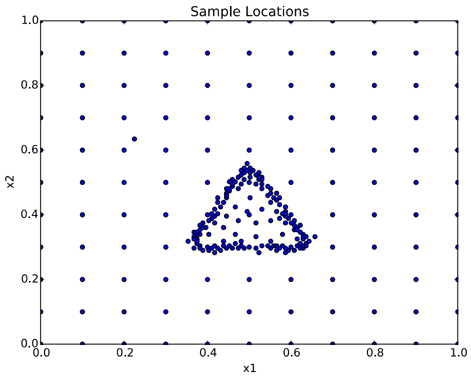
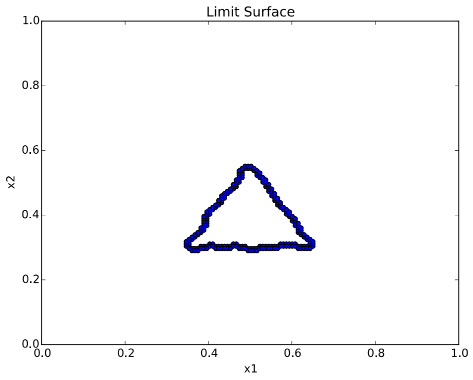
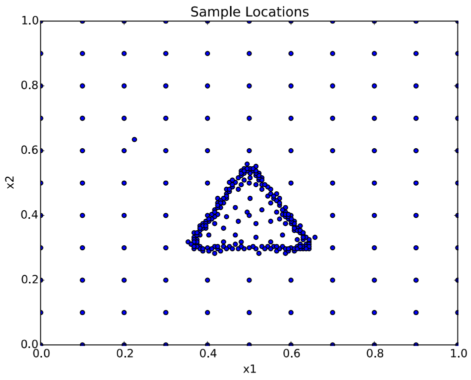
Iteration	Estimated limit surface	Sample locations
50		
100		
150		
199		

Figure 41 – SVM: calculation evolution of the triangular limit surface

```
<Convergence limit = "3000" weight = "value" persistence = "0">5.0e-
5</Convergence>
```

RAVEN Input block 4 – Convergence settings for the triangular test: SVM case

After an initial grid of 11 × 11, convergence was obtained after 199 samples. The obtained expected value was 9.05592 E-01. Figure 40 and Figure 41 shows the evolution of the estimate of the limit surface and the locations of the samples throughout the adaptive sampling process.

9.1.3 TMI SBO Test

For the TMI SBO test, we used SVM using Gaussian Kernel as indicated below.

```
<ROM name="AccelerationROM" subType="SciKitLearn">
  <Features>x1,x2</Features>
  <Target>goalFunction</Target>
  <SKLtype>svm|SVC</SKLtype>
  <kernel>rbf</kernel>
  <gamma>10.0</gamma>
  <C>0.1</C>
</ROM>
```

RAVEN Input block 5 – SVM definition for the triangular test

```
<Convergence limit = "3000" weight = "value" persistence = "0">5.0e-5</Convergence>
```

RAVEN Input block 6 – Convergence settings for the TIM SBO test: SVM case

After an initial grid of 6x6, convergence was obtained after 270 samples. The obtained expected value was 2.5762 E-1 (theoretical value equal to 2.57531 E-1). Figure 42 and Figure 43 show the evolution of the estimate of the limit surface and the locations of the samples throughout the adaptive sampling process.

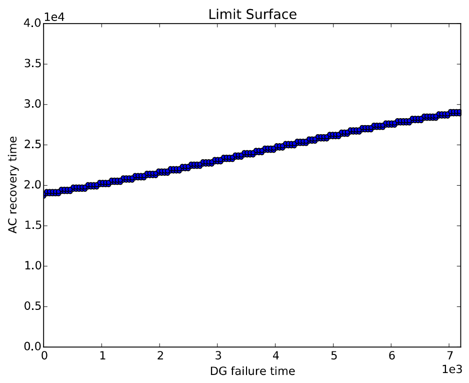
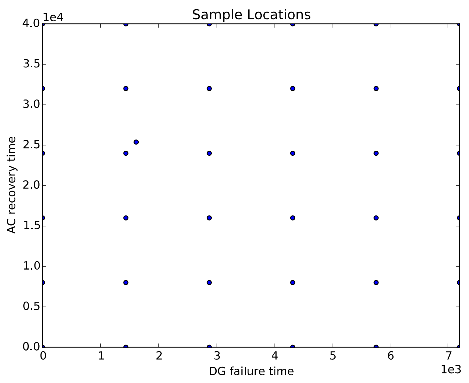
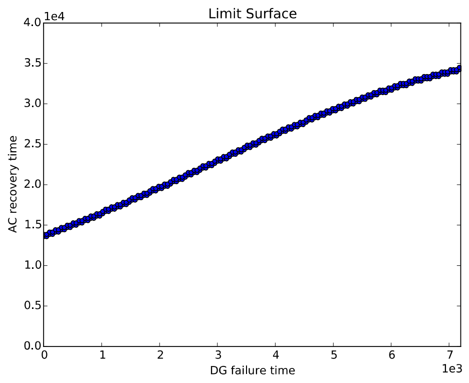
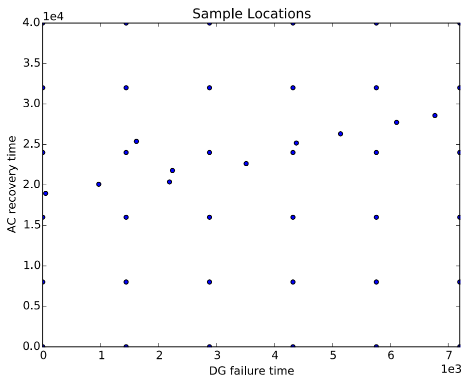
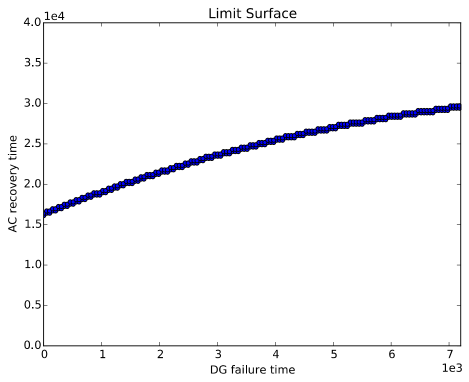
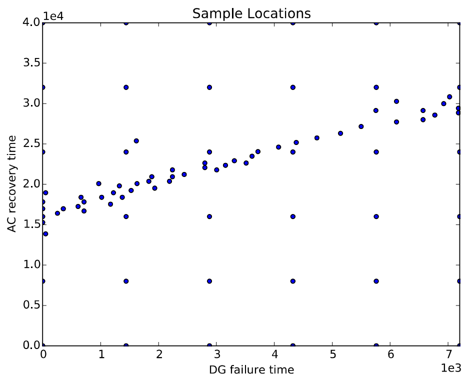
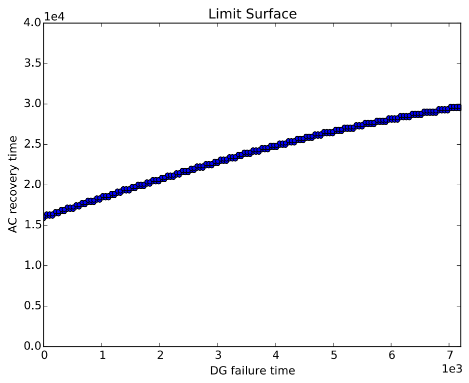
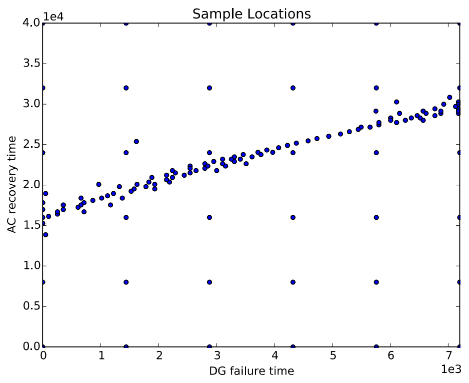
Iteration	Estimated limit surface	Sample locations
1		
10		
50		
100		

Figure 42 – SVM: calculation evolution of the TMI SBO limit surface

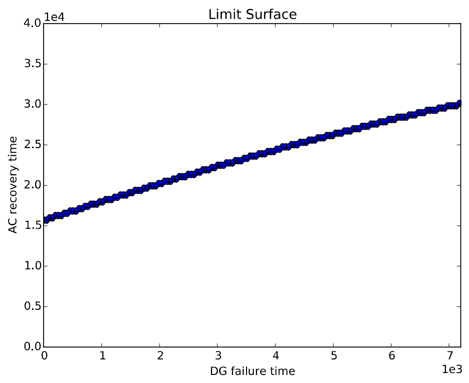
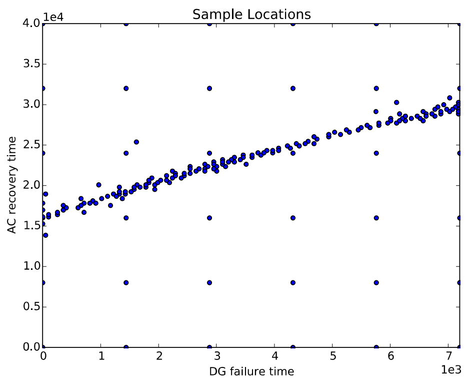
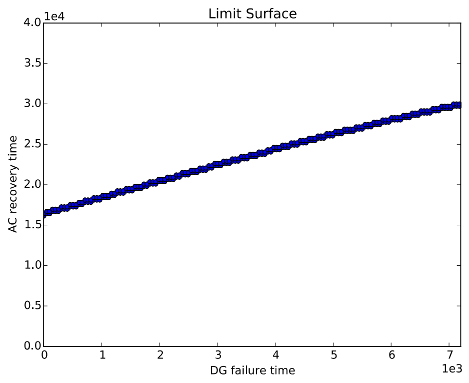
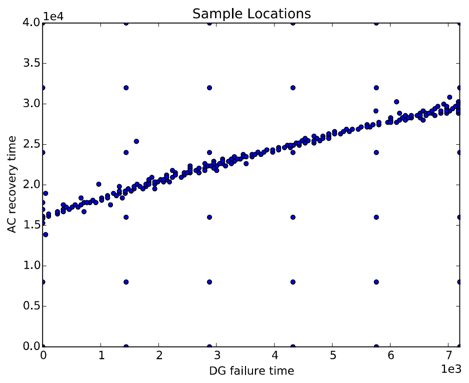
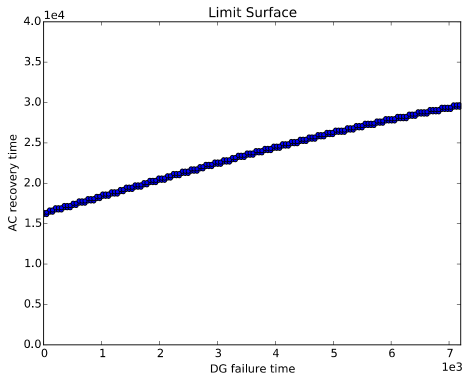
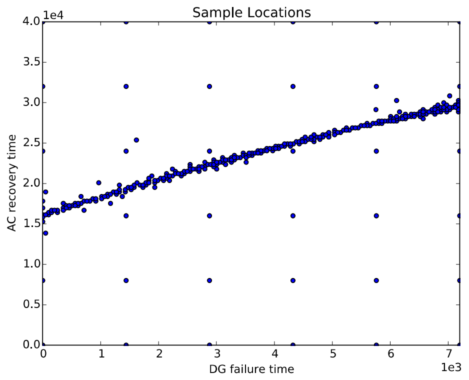
Iteration	Estimated limit surface	Sample locations
150		
200		
270		

Figure 43 – SVM: calculation evolution of the TMI SBO limit surface

9.1.4 Convergence study on SVM

We performed a series of convergence studies to test how SVM performs using the TMI SBO test case for different values of Gamma and C. We were interested into observing the sample locations and the relative error into the evaluation of the limit surface.

For all cases we set the adaptive sampling convergence criteria as follows:

```
<Convergence limit = "3000" weight = "value" persistence = "0">5.0e-5</Convergence>
```

RAVEN Input block 7 – Convergence settings for the triangular test: SVM case

Figure 45 and Figure 44 show the sample locations for Gamma = 1.0 and Gamma =10.0 for several values of C (0.1, 1.0, 10.0, 100.0 and 1000.0). All cases started from an initial set of training points on a 6x6 grid. Note how by increasing the value of C, fewer degrees of freedom are given to the limit surface to change and thus, fewer samples are needed.

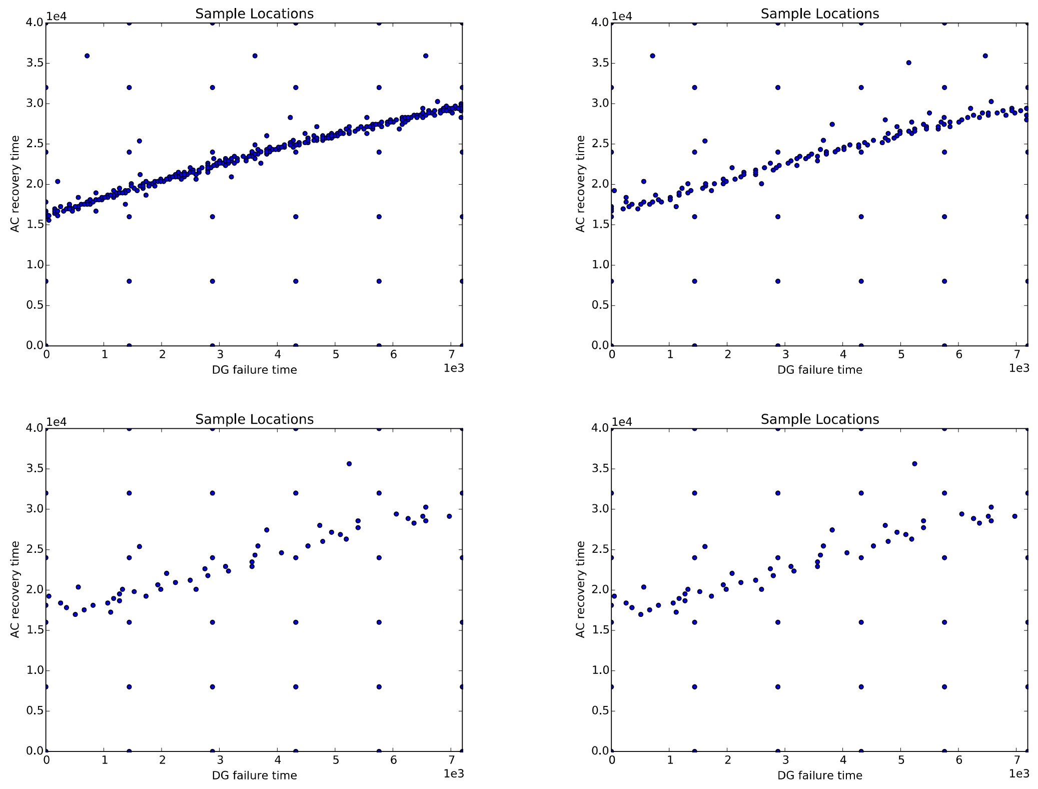


Figure 44 – Sample locations for TMI SBO limit surface using SVM with gamma = 10.0: C= 1.0 (top left), 10.0 (top right), 100.0 (bottom left) and 1000.0 (bottom right)

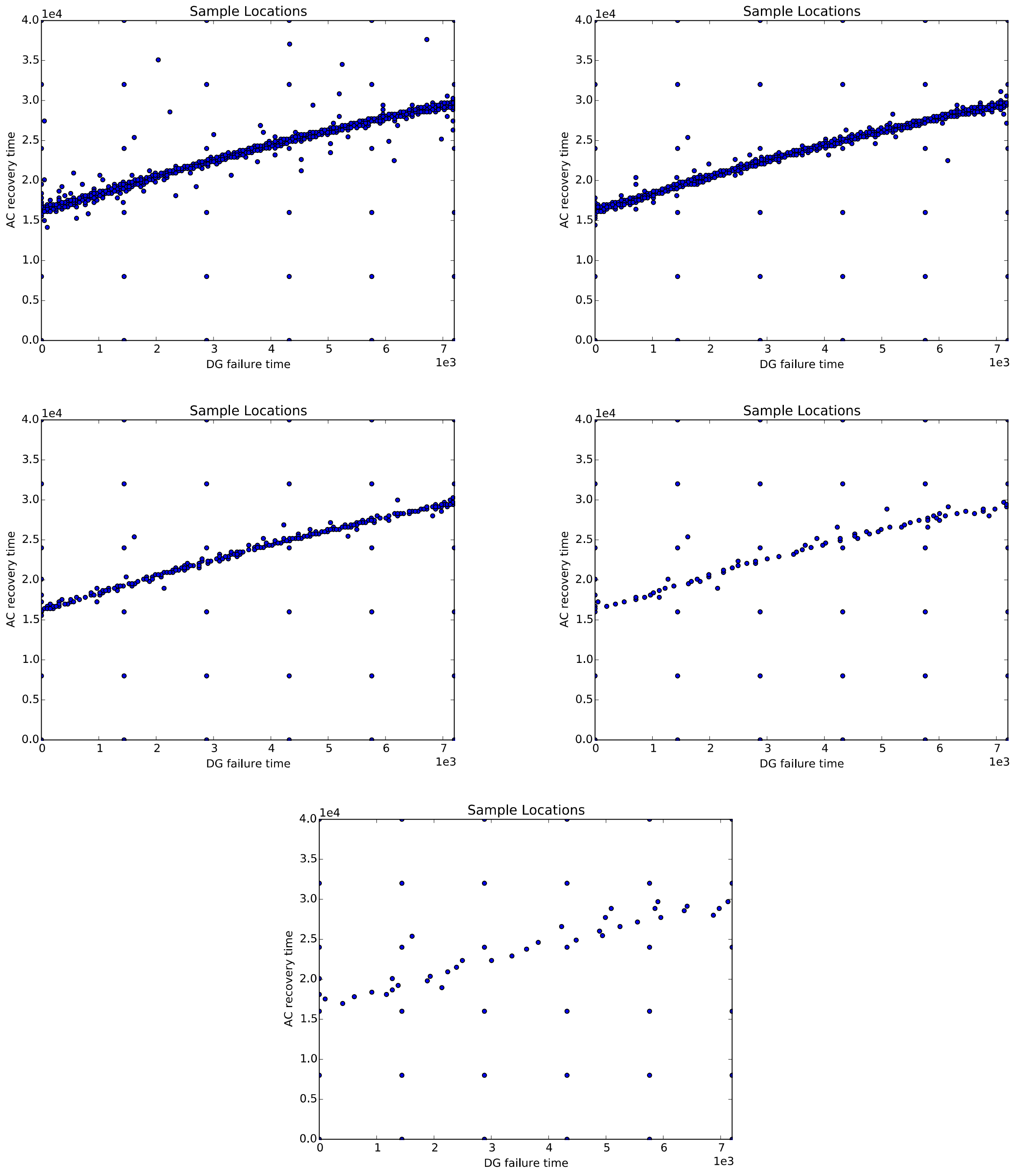


Figure 45 – Sample locations for TMI SBO limit surface using SVM with Gamma = 1.0: C= 0.1 (top left), 1.0 (top right), 10.0 (mid left), 100.0 (mid right) and 1000.0 (bottom)

We additionally performed a more exhaustive testing by measuring the relative error associated to the estimate of the limit surface again by using the TMI SBO test case. We measured such relative error for several values of both Gamma and C; this is shown in Figure 46. Note that not all combinations of gamma and C values are possible.

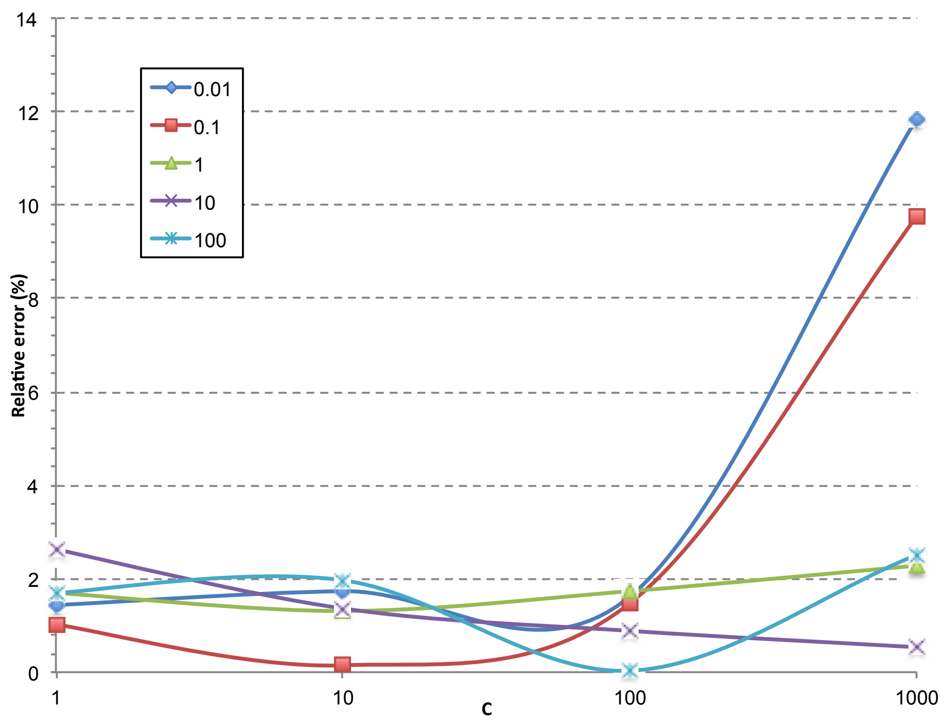


Figure 46 – SVM: plot of max relative error as functions of *C* and *gamma*

9.2 KNN Classifier

The identical tests performed in Section 9.1 for the SVM classifier were performed also for the KNN classifiers. The results are shown in the next four sections

9.2.1 Rosenbrock Test

Regarding the Rosenbrock test we chose a KNN algorithms with a $K = 5$ as shown below:

```
<ROM name="AccelerationROM" subType="SciKitLearn">
   <Features>x1,x2</Features>
   <Target>goalFunction</Target>
   <SKLtype>neighbors|KNeighborsClassifier</SKLtype>
   <n neighbors>5</n neighbors>
</ROM>
```

RAVEN Input block 8 – KNN definition for the Rosenbrock test

Convergence was obtained after 675 samples and the expected value was 2.21889 E-01.

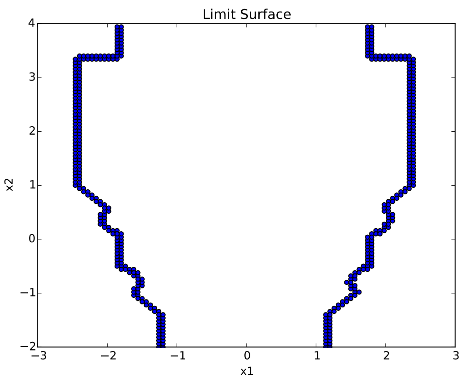
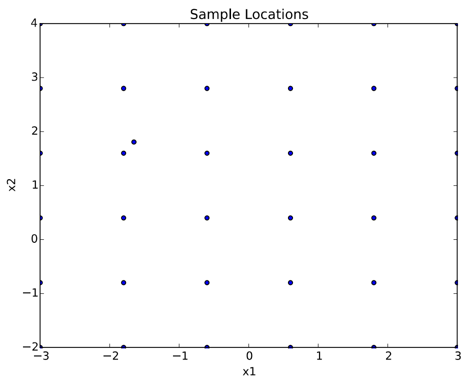
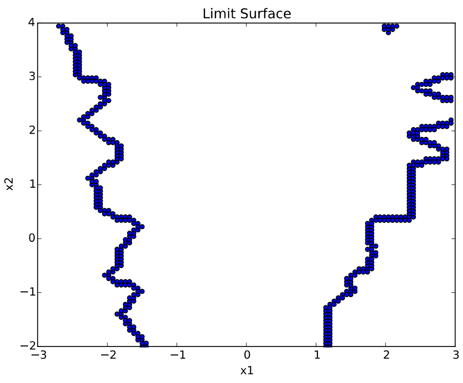
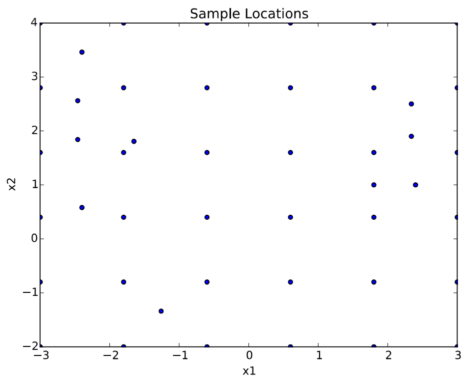
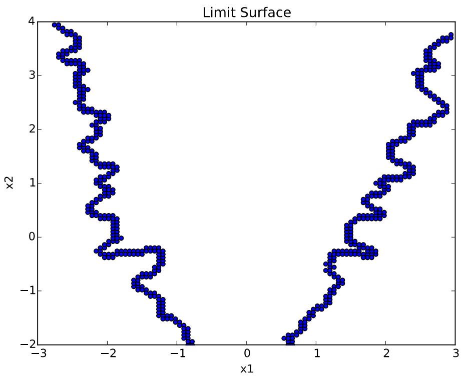
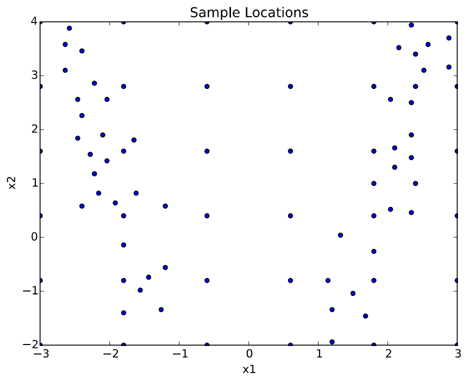
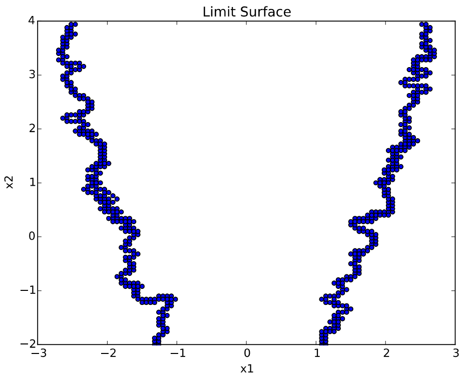
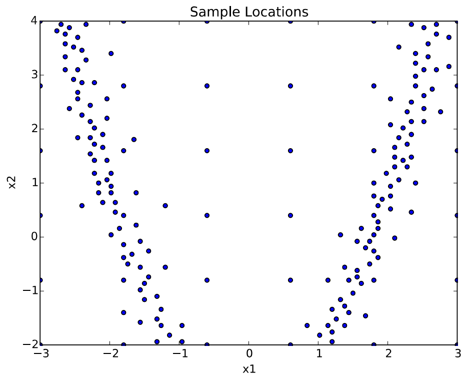
Iteration	Estimated limit surface	Sample locations
1		
10		
50		
150		

Figure 47 – KNN: calculation evolution of the Rosenbrock limit surface

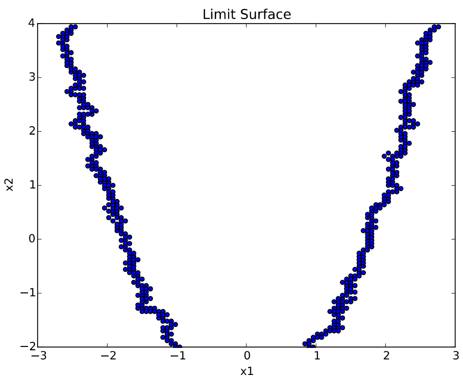
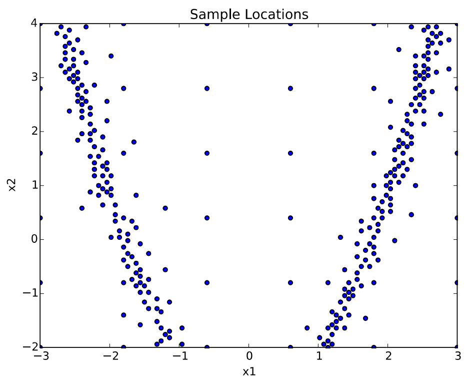
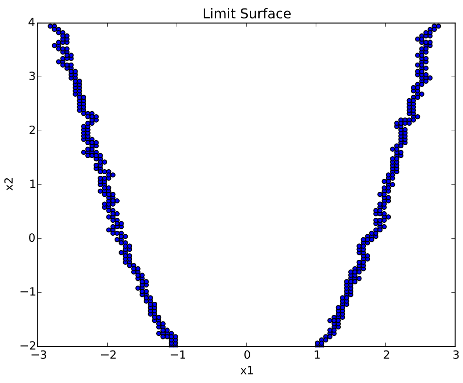
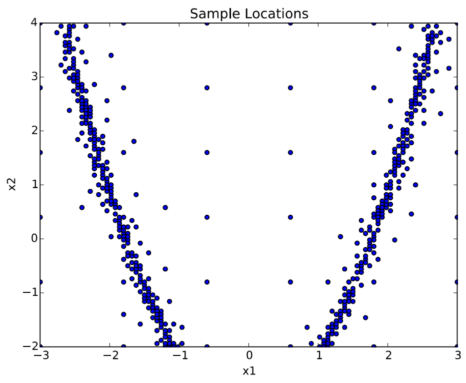
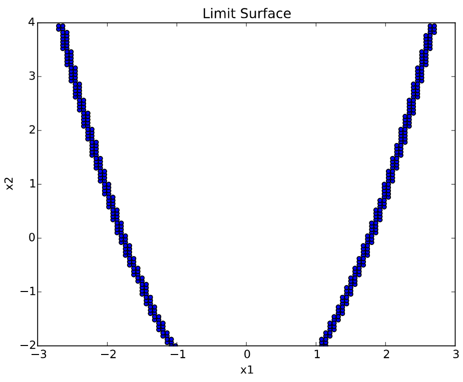
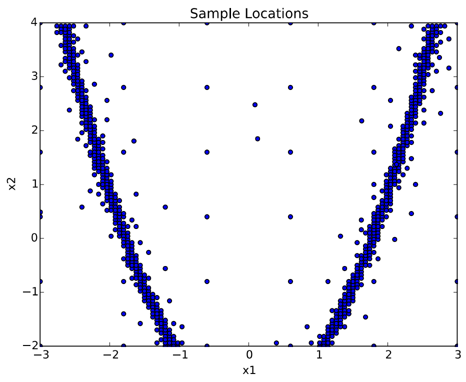
Iteration	Estimated limit surface	Sample locations
250		
450		
675		

Figure 48 – KNN: calculation evolution of the Rosenbrock limit surface

9.2.2 Triangular Test

Regarding the Triangular test we chose a KNN algorithm with a lower value of $K = 2$ as shown below. This was due to the fact that a lower value of K would increase the performances of the algorithm.

```
<ROM name="AccelerationROM" subType="SciKitLearn">
   <Features>x1,x2</Features>
   <Target>goalFunction</Target>
   <SKLtype>neighbors|KNeighborsClassifier</SKLtype>
   <n_neighbors>2</n_neighbors>
</ROM>
```

RAVEN Input block 9 – KNN definition for the Rosenbrock test

After an initial grid 11×11, convergence was obtained after 556 samples and the expected value was 9.04178 E-01 as shown in Figure 49 and Figure 50.

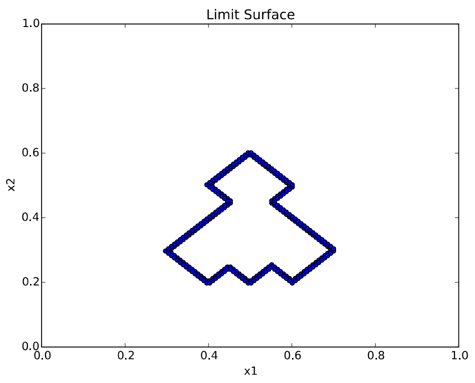
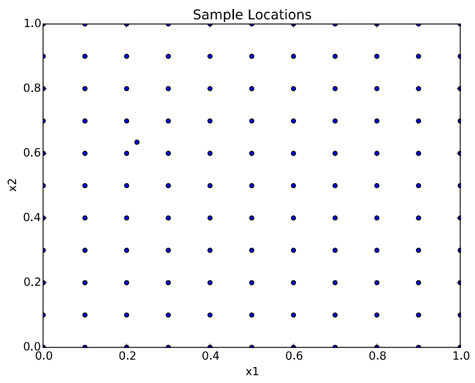
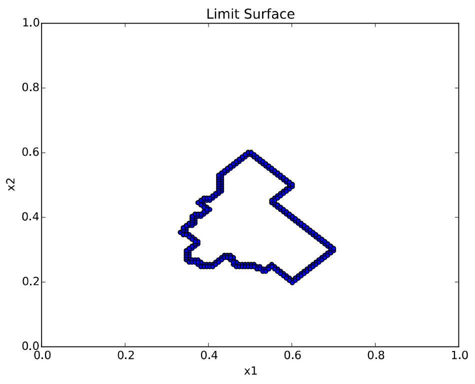
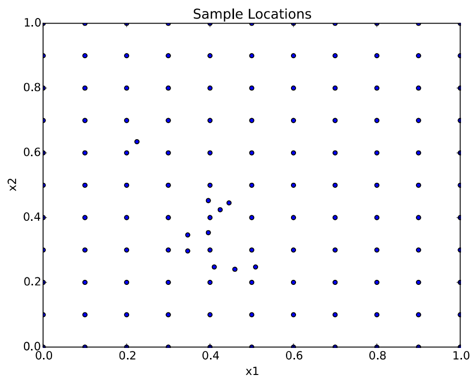
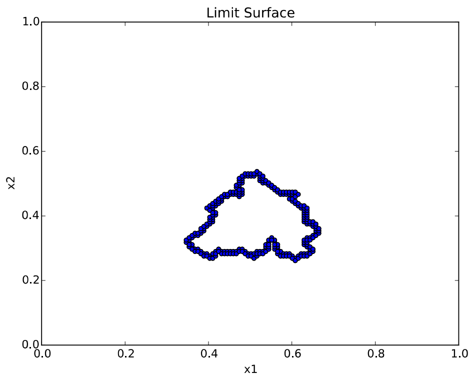
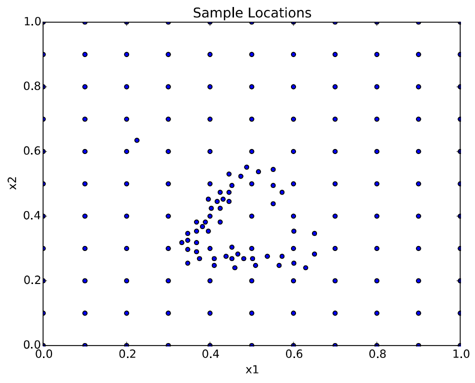
Iteration	Estimated limit surface	Sample locations
1		
10		
50		

Figure 49 – KNN: calculation evolution of the Triangular limit surface

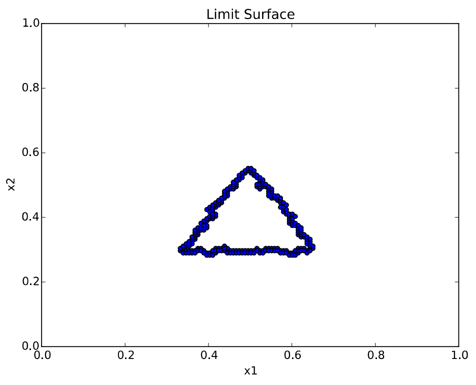
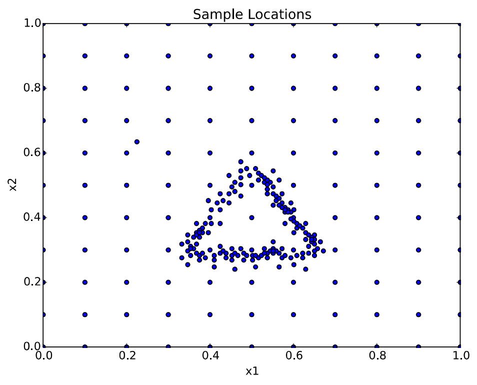
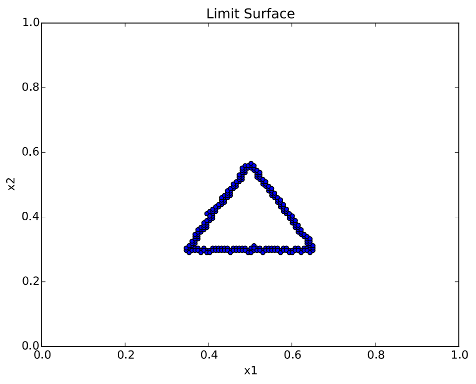
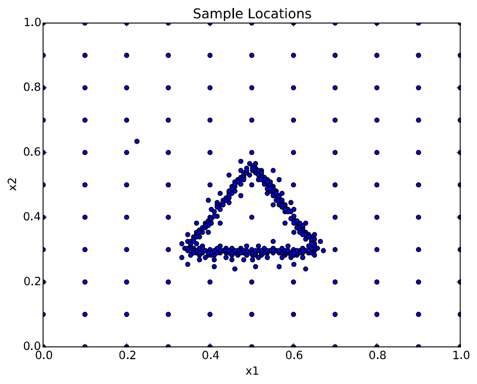
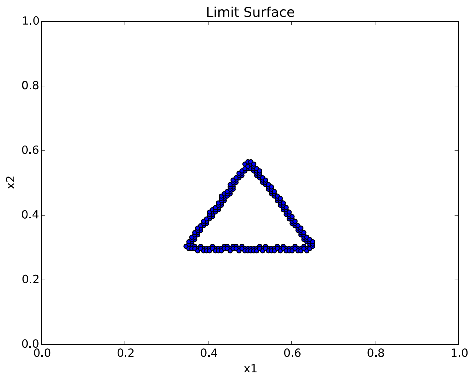
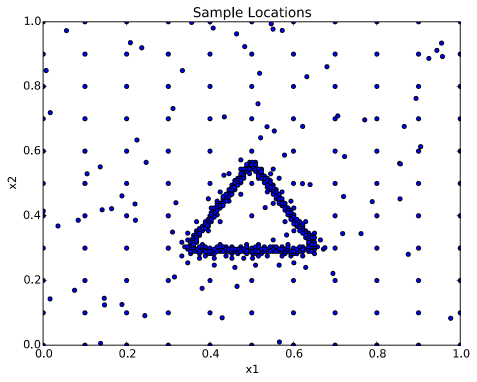
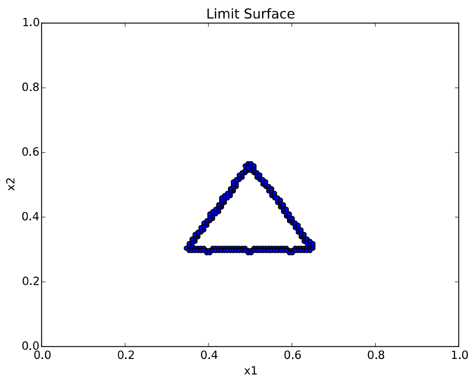
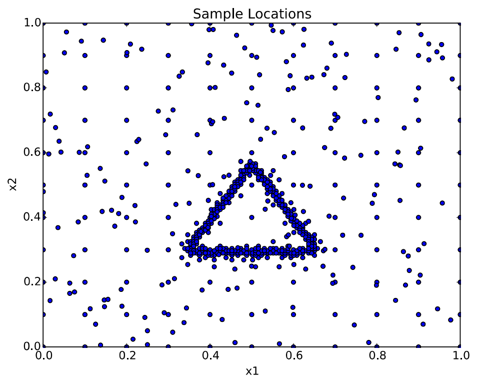
Iteration	Estimated limit surface	Sample locations
150		
300		
450		
556		

Figure 50 – KNN: calculation evolution of the triangular limit surface

9.2.3 TMI SBO Test

Regarding the TMI SBO test we chose a KNN algorithm with a value of *K* = 5 as shown below:

```
<ROM name="AccelerationROM" subType="SciKitLearn">
   <Features>x1,x2</Features>
   <Target>goalFunction</Target>
   <SKLtype>neighbors|KNeighborsClassifier</SKLtype>
   <n_neighbors>5</n_neighbors>
</ROM>
```

RAVEN Input block 10 – KNN definition for the TMI SBO test

After an initial grid 6 × 6, convergence was obtained after 328 samples; expected value was 2.56732 E-1.

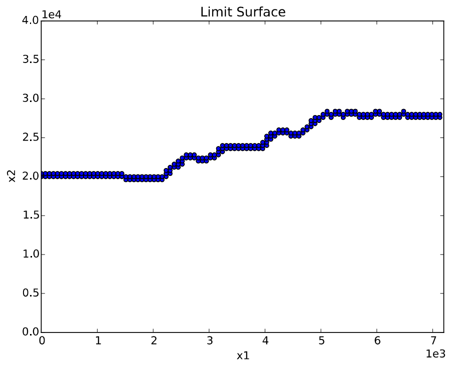
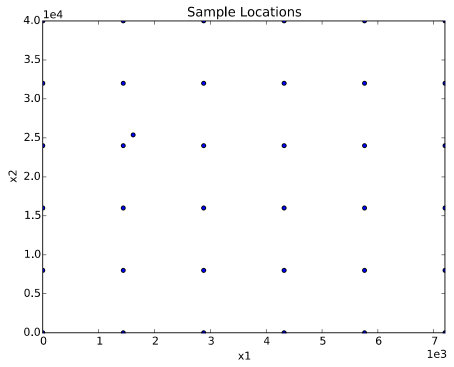
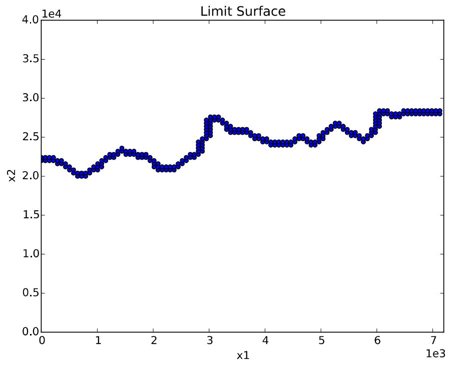
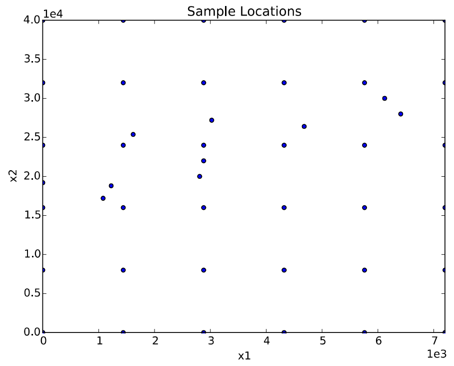
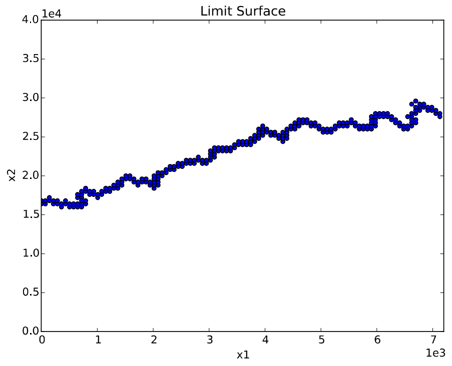
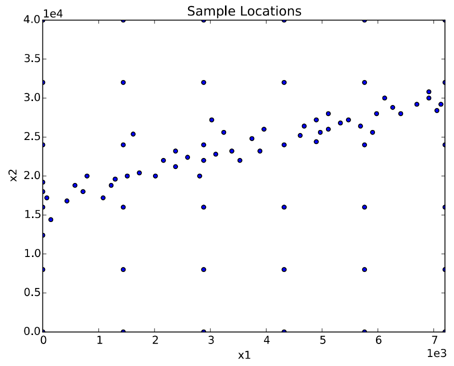
Iteration	Estimated limit surface	Sample locations
1		
10		
50		

Figure 51 – KNN: calculation evolution of the TMI SBO limit surface

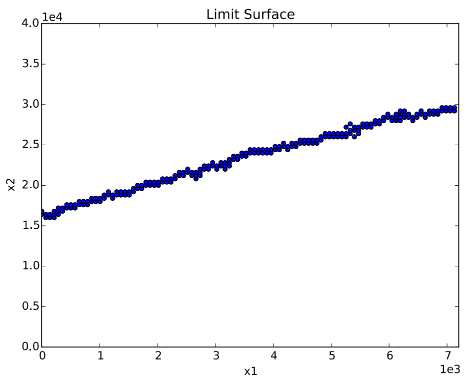
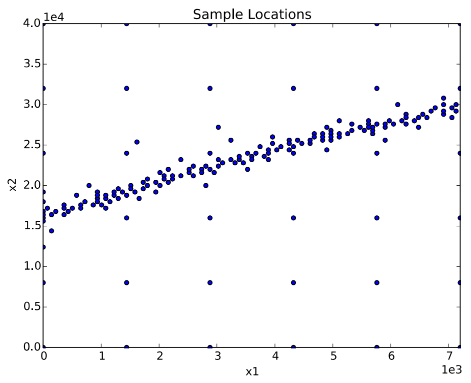
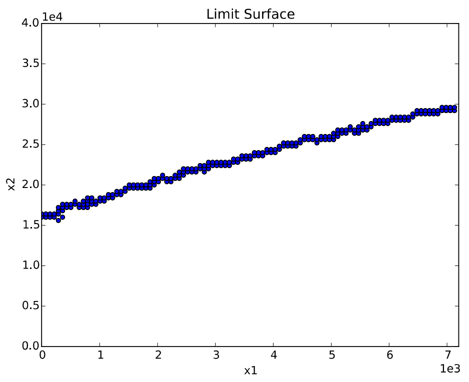
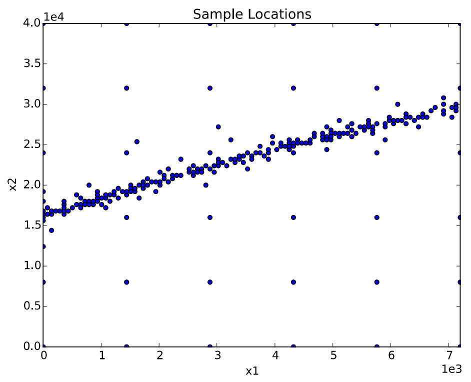
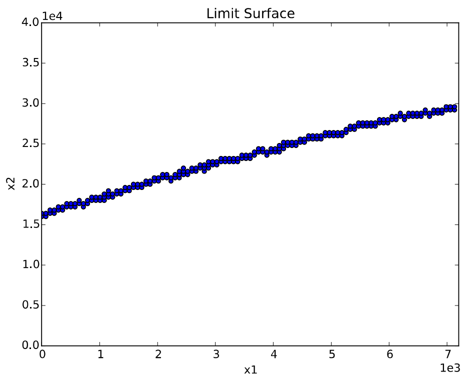
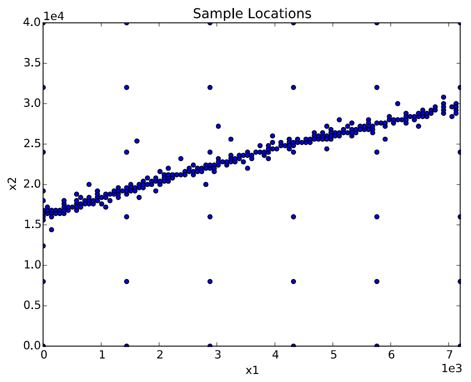
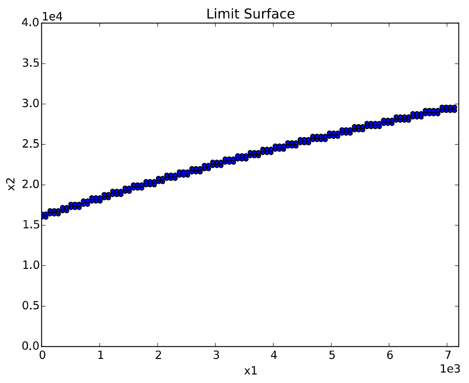
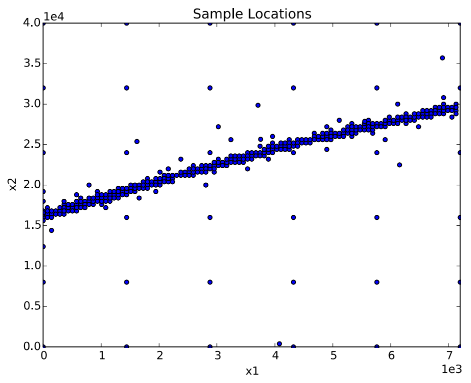
Iteration	Estimated limit surface	Sample locations
150		
200		
250		
328		

Figure 52 – KNN: calculation evolution of the TMI SBO limit surface

9.2.4 Convergence study on KNN

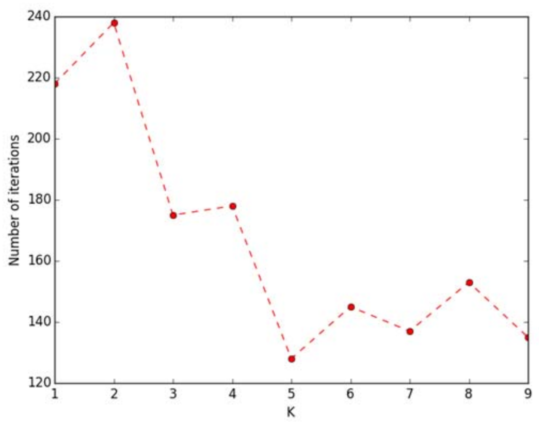
The convergence study on the KNN algorithm was performed by changing the value of *K*. The relative error in the evaluation of the limit surface and the number of samples required are summarized in Table 5 and pictured in Figure 53.

```
<Convergence limit="3000" weight="value" persistence="0">1.0e-4</Convergence>
```

RAVEN 11 – Convergence settings for the TMI SBO test: KNN case

Table 5 – KNN classifier: number of iterations and relative error for the TMI SBO test case as function of K

K	# iterations	Relative error (%)
1	218	0.392574098
2	238	0.20347065
3	175	0.480330523
4	178	0.781653471
5	128	0.249290377
6	145	0.050867663
7	137	1.384687669
8	153	0.458973871
9	135	0.376653684



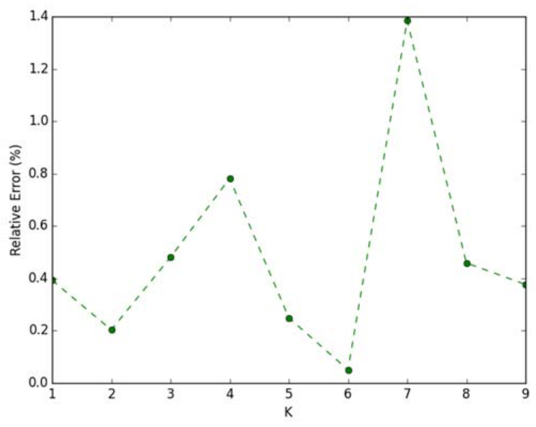


Figure 53 – KNN classifier: plot of the number of iterations and relative error for the TMI SBO test case as function of K

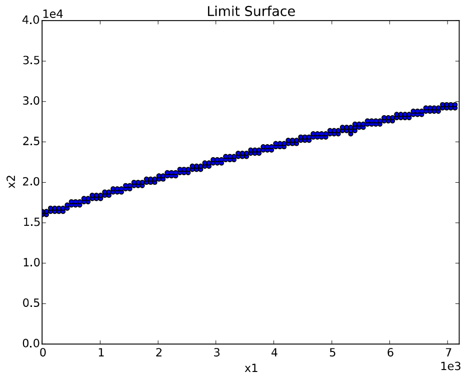
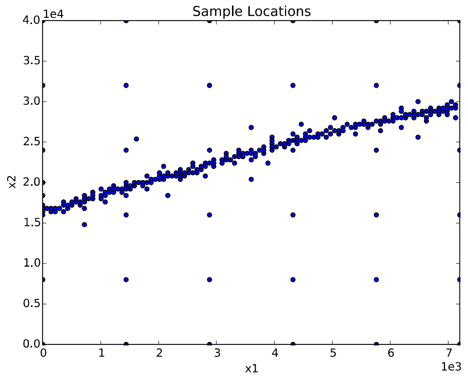
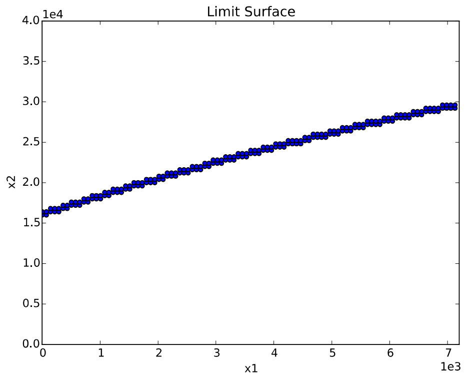
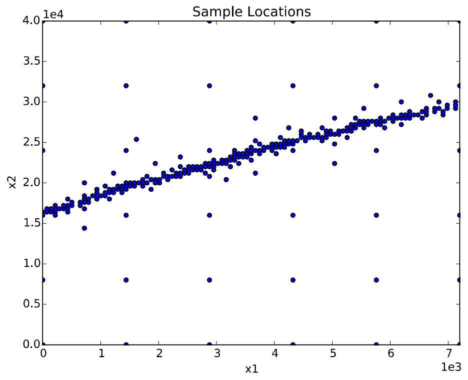
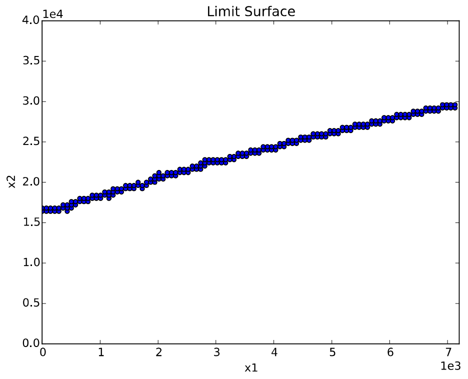
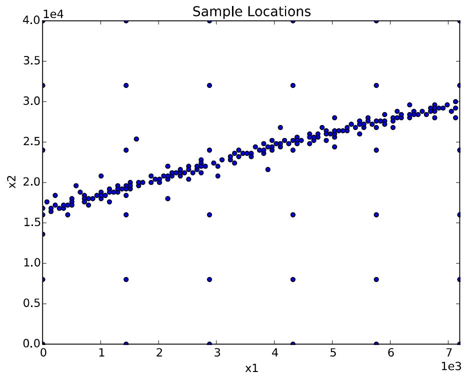
K	Estimated limit surface	Sample locations
1		
2		
3		

Figure 54 – KNN classifier: plot of the limit surface and the sample locations for the TMI SBO test case as function of k

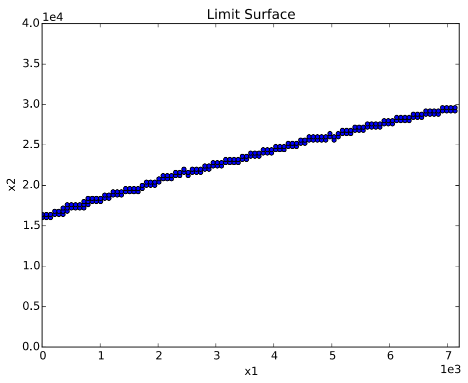
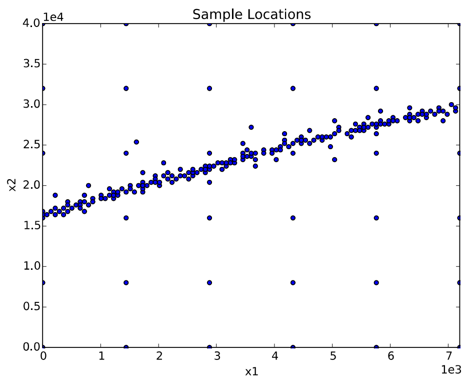
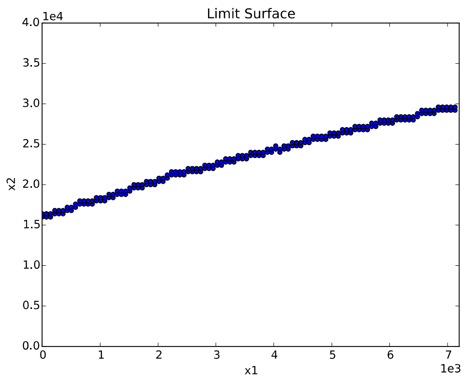
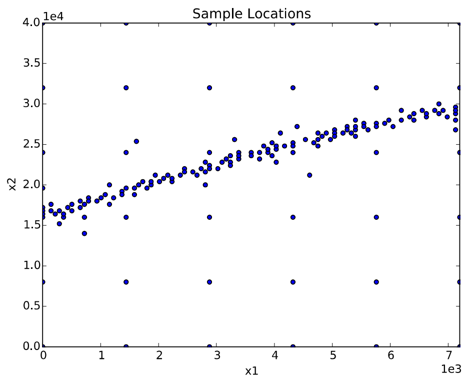
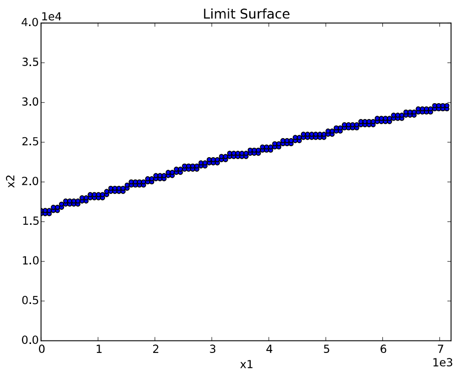
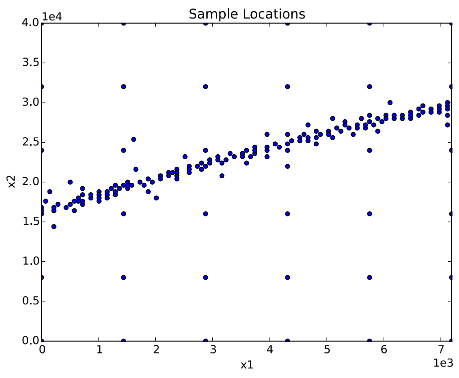
K	Estimated limit surface	Sample locations
4		
5		
6		

Figure 55 – KNN classifier: plot of the limit surface and the sample locations for the TMI SBO test case as function of k

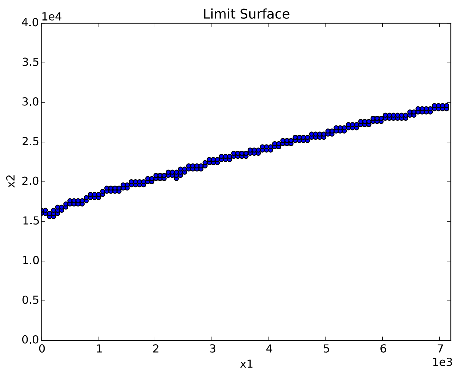
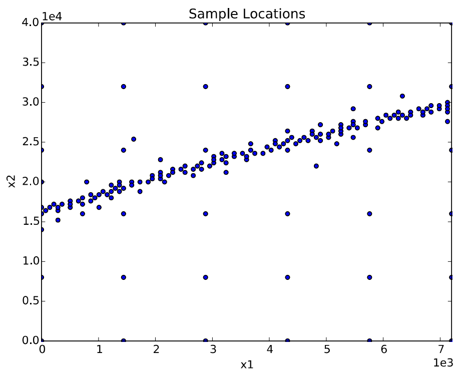
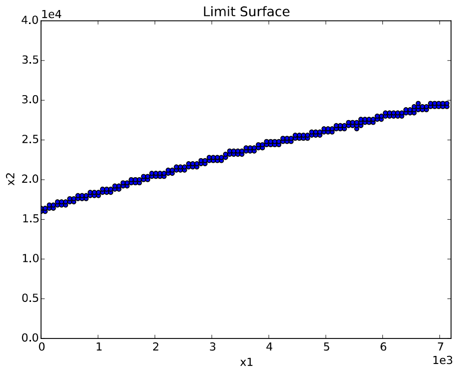
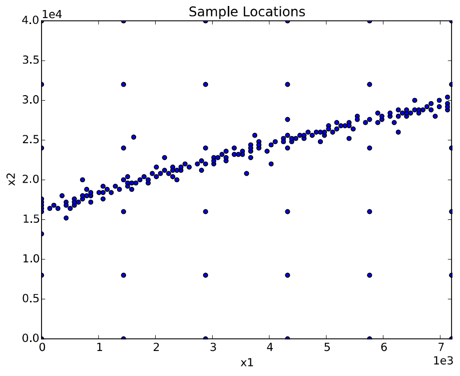
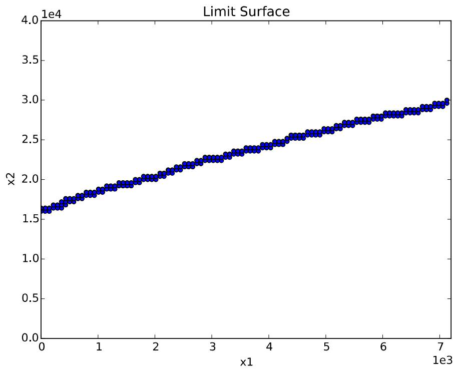
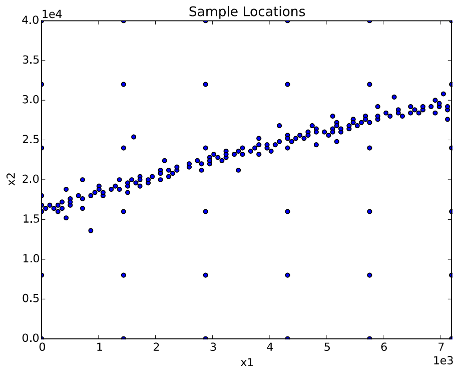
K	Estimated limit surface	Sample locations
7		
8		
9		

Figure 56 – KNN classifier: plot of the limit surface and the sample locations for the TMI SBO test case as function of K

10. COMPARISON AND CONVERGENCE STUDIES OF REGRESSORS

Regarding the set of regressors available in RAVEN we chose a subset of them: GPM, KNR, SVR, Sheppard and gPC. For all of them we performed a series of test case using the Rosenbrock function, the Paraboloid and the RELAP-7 PWR natural circulation loop and a set of convergence studies. A map of the structure of this report is shown below.

	Test Case			Convergence Study
	Rosenbrock	Paraboloid	RELAP-7 loop	
GPM	Section 10.1.1	Section 10.1.2	Section 10.1.3	Section 10.1.4-5
KNR	Section 10.2.1	Section 10.2.2	Section 10.2.3	Section 10.2.4
SVR	Section 10.3.1	Section 10.3.2	Section 10.3.3	Section 10.3.4
Sheppard	Section 10.4.1	Section 10.4.2	Section 10.4.3	Section 10.4.4
gPC	Section 10.5.1	Section 10.5.2	Section 10.5.3	Section 10.5.4

10.1 Gaussian Process Models

10.1.1 Rosenbrock Test

We tested the GPM surrogate model the Rosenbrock function starting from a 6 × 6 grid training set as shown below:

```
<ROM name="ROM_GPM" subType="SciKitLearn">
   <SKLtype>GaussianProcess|GaussianProcess</SKLtype>
   <Features>x1,x2</Features>
   <Target>y1</Target>
   <theta0>1e-2</theta0>
   <thetaL>1e-4</thetaL>
   <thetaU>1e-1</thetaU>
   <nugget>0.00000000000001</nugget>
</ROM>
```

RAVEN Input block 12 – GPM definition for the Rosenbrock function test

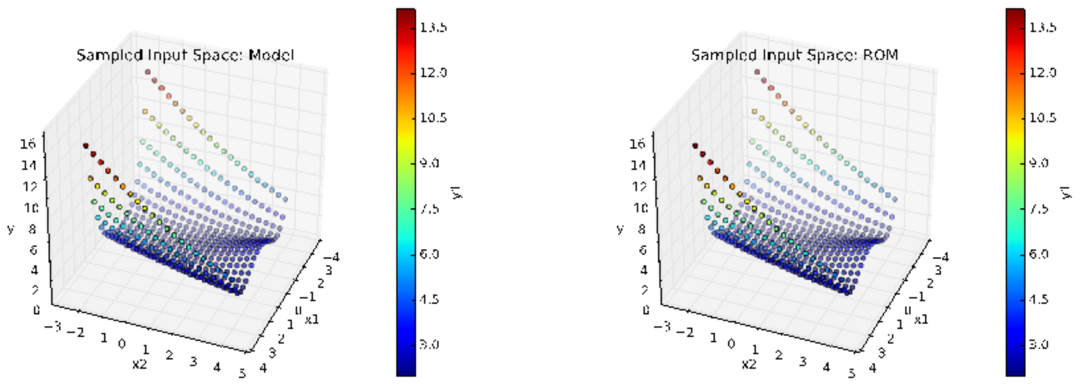


Figure 57 – GPM: Comparison of the response function for the Rosenbrock test case

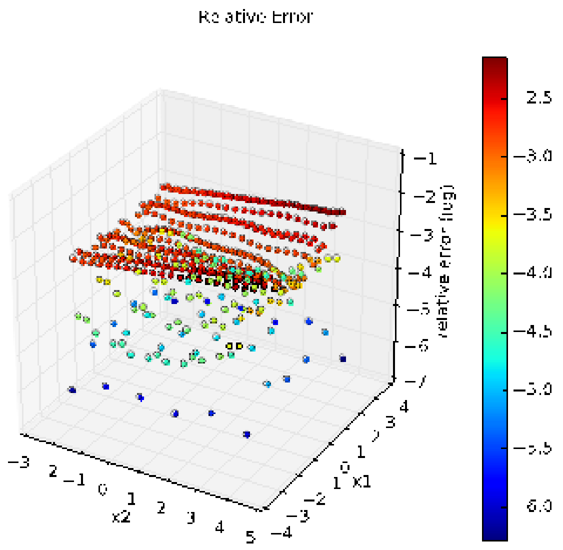


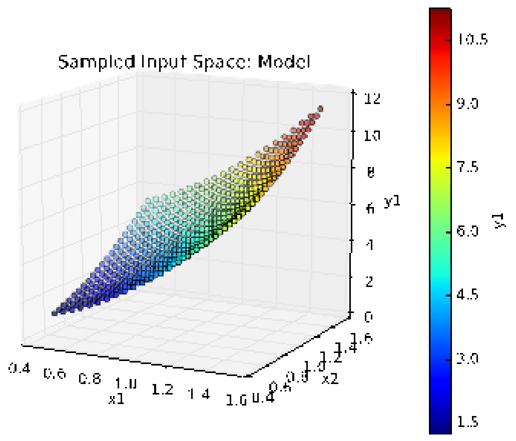
Figure 58 – GPM: relative error for the Rosenbrock test case

Regression results are shown in Figure 57 and Figure 58. Note from Figure 58, the low relative error which has a maximum value equal to 0.007068.

10.1.2 Paraboloid test

The scope of the Paraboloid test case is to compare not the regression error in the reconstruction of the response surface but to evaluate the impact of such error when UQ and SA analyses are performed. In Figure 59 the theoretical and the reconstructed functions are shown. The regression is performed from an initial grid 6 × 6.

Table 6 compares the first three statistical moments, the sensitivity and the Pearson coefficients obtained from the GPM surrogate model and the theoretical ones. In addition, Figure 60 shows a comparison of the distributions of y1. Note how the low relative error in the regression process strongly reduces the error in the evaluation of the statistical moments.



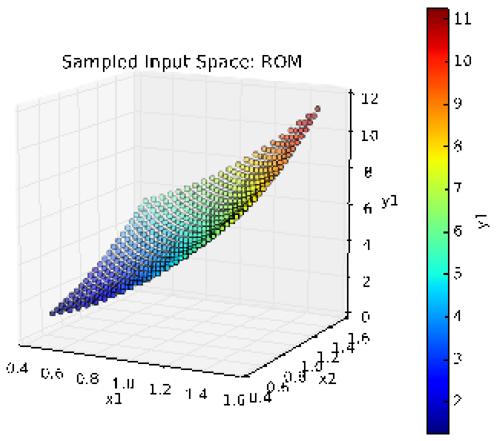
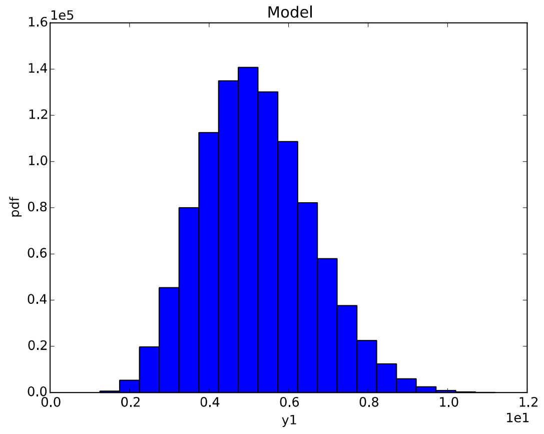


Figure 59 – GPM: Comparison of the response function for the paraboloid test case

Table 6 – GPM: Comparison of the statistical moments, Pearson and sensitivity coefficients for y1 for the paraboloid test case

	Reference	ROM	Relative Error
mu	5.18212435	5.18212456	-4.05239 E-08
sigma	1.38749472	1.38749458	1.00901 E-07
skewness	0.339384699	0.339384298	1.18155 E-06
x1 sensitivity	5.99808493	5.99808449	7.33567 E-08
x2 sensitivity	4.00024747	4.00024718	7.24955 E-08
x1 Pearson	0.825704716	0.825704743	-3.26993 E-08
x2 Pearson	0.552788789	0.552788807	-3.25622E-08



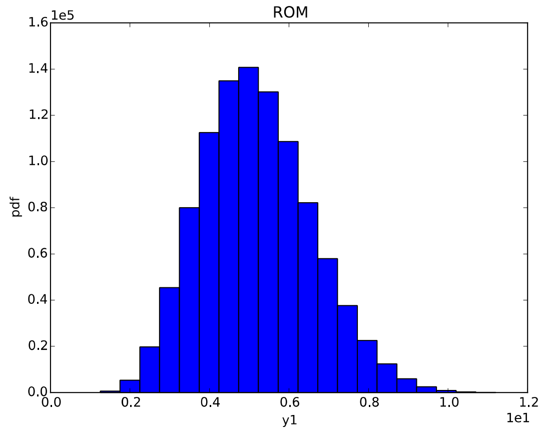


Figure 60 – GPM: Comparison of the pdfs of y1 for the paraboloid test case

10.1.3 PWR Natural Circulation Loop test

We performed the same analysis shown in Section 10.1.2 but using the RELAP-7 PWR natural circulation loop and the GPM surrogate model as shown below (initial training grid was 6 × 6):

```
<ROM name="ROM_GPM" subType="SciKitLearn">
   <SKLtype>GaussianProcess|GaussianProcess</SKLtype>
   <Features>Materials|fuel-mat|k,Components|reactor|power</Features>
   <Target>PeakFuelTemperature</Target>
   <theta0>1e-2</theta0>
   <thetaL>1e-4</thetaL>
   <thetaU>1e-1</thetaU>
   <nugget>0.00000000000001</nugget>
</ROM>
```

RAVEN Input block 13 – GPM definition for the Rosenbrock PWR Natural Circulation Loop test

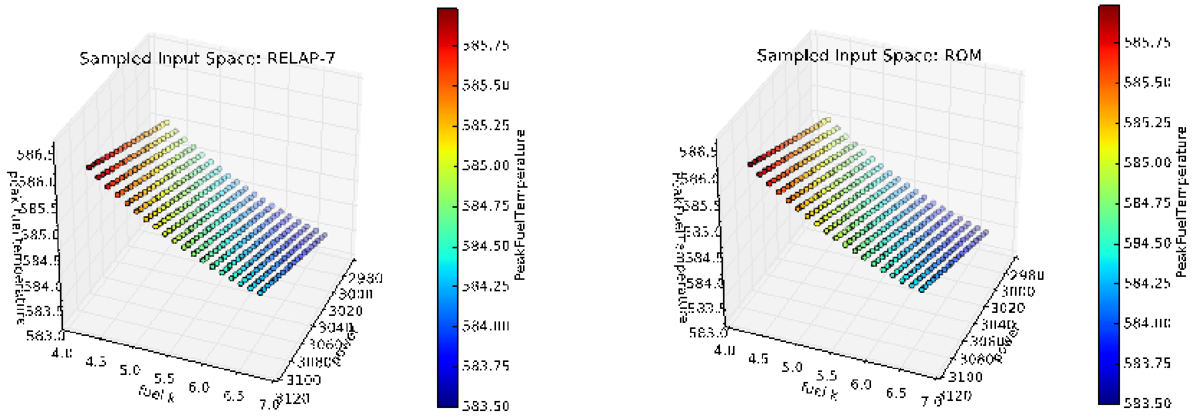


Figure 61 – GPM: Comparison of the response function for the PWR natural circulation loop test case

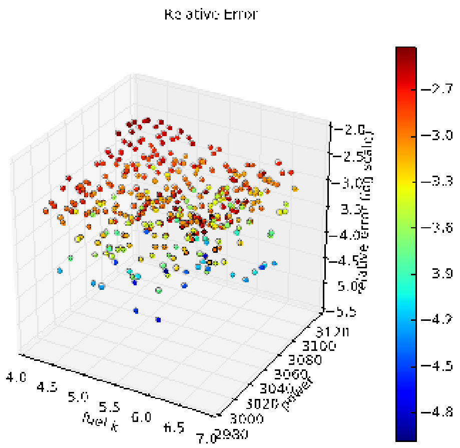


Figure 62 – GPM: Relative error response function for the PWR natural circulation loop test case

The response functions comparison is shown in Figure 62 while the plot of the relative error is shown in Figure 63 (the maximum relative error was 0.0037668). Table 7 summarizes the comparison of the statistical moments and sensitivity coefficients. Figure 63 shows the plot of the obtain distribution of max fuel temperature given the uncertainties of the two input parameters.

Table 7 – GPM: Comparison of the statistical moments, Pearson and sensitivity coefficients for y1 for the PWR natural circulation loop test case

	RELAP-7	ROM	Relative Error
mu	5.84550604 E+02	5.84550376 E+02	3.90043 E-07
sigma	3.46907782 E-01	3.47242245 E-01	-0.000964127
skewness	3.44692439 E-01	3.43410411 E-01	0.003719339
K sensitivity	-6.66368173 E-01	-6.67137762 E-01	-0.001154901
P sensitivity	7.23160579 E-03	7.23424644 E-03	-0.000365154
K pearson	-9.05607661 E-01	-9.05780132 E-01	-0.000190448
P pearson	-3.97276934 E-04	-3.97276934 E-04	0

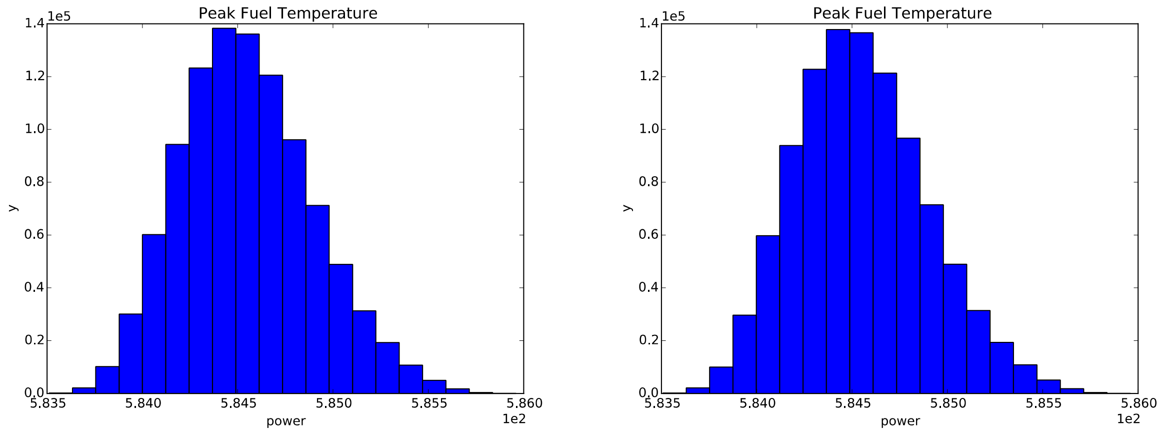


Figure 63 – GPM: Comparison of the pdfs of peak fuel temperature for the PWR natural circulation loop test case

10.1.4 Convergence Study on GPMs: Initial Grid

The first convergence study was performed by changing the number of samples used in training phase. The rationale is that an increasing number of training samples decreases the max relative error in the reconstruction of the response function.

```
    <ROM name="ROM_GPM" subType="SciKitLearn">
        <SKLtype>GaussianProcess|GaussianProcess</SKLtype>
        <Features>Materials|fuel-mat|k,Components|reactor|power</Features>
        <Target>PeakFuelTemperature</Target>
        <theta0>1e-2</theta0>
        <thetaL>1e-4</thetaL>
        <thetaU>1e-1</thetaU>
        <nugget>0.00000000000001</nugget>
    </ROM>
```

RAVEN Input block 14 – GPM definition for the convergence study on the initial grid

As predicted the results are shown in Table 8 and they are plotted in Figure 64.

Table 8 – GPM: Max relative error as function of the initial grid size for the Rosenbrock test case

Grid size	Relative error
3	0.776149657892
5	0.00706802106224
7	0.000239270887758
9	8.7723003133 E-05
11	1.8937001264 E-05
14	6.62039016827 E-06

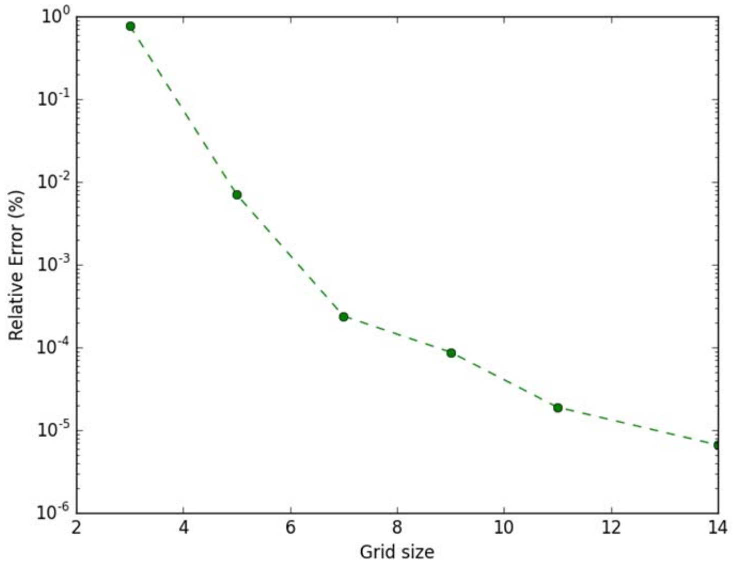


Figure 64 – GPM: Relative error for the Rosenbrock test case as function of the size of the initial grid

10.1.5 Convergence Study on GPMs: Internal parameters

The second convergence study focuses on the internal parameters of the GPM surrogate model. For the scope of this report we focused only on the parameter *theta*. The obtained results are shown in Figure 65 and summarized in Figure 66. Note that a higher value of *theta* forces the obtained reconstruction to be smoother, i.e., with less degree of freedoms.

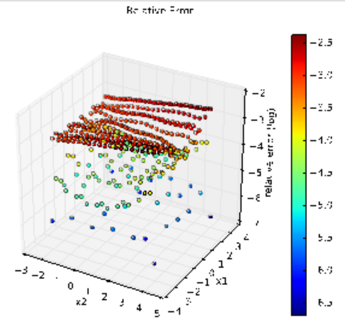
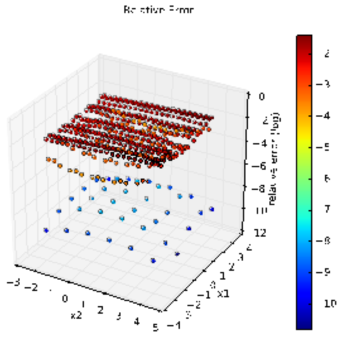
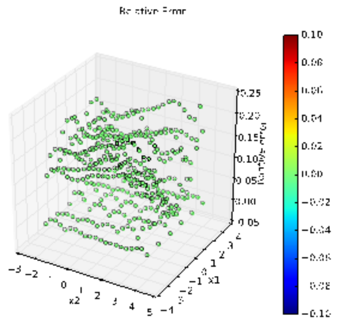
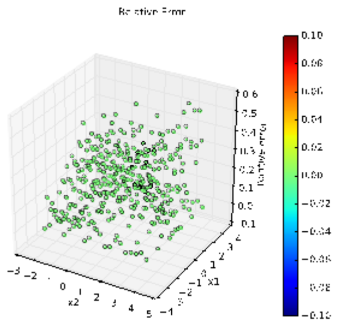
Theta	Max Relative error	Relative error
0.01	0.00426246061374	
0.1	0.0395741607288	
1	0.232405124639	
10	0.566013953719	

Figure 65 – GPM: max value and plot of the relative error as function of theta for the Rosenbrock test case

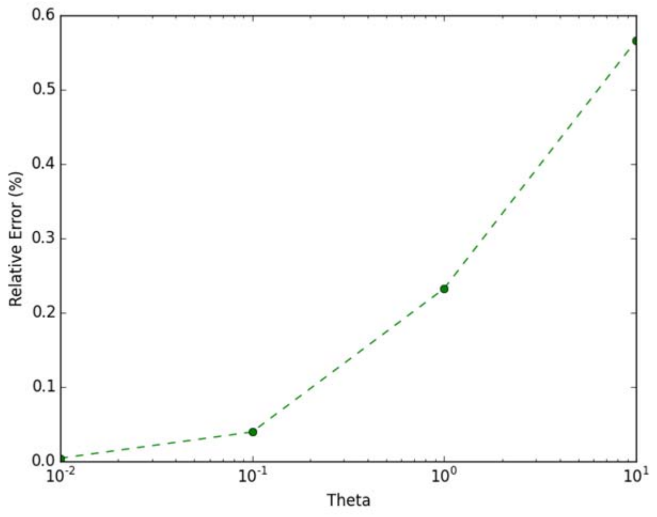


Figure 66 – GPM: max relative error as function of theta for the Rosenbrock test case

10.2 K-Neighbors Regressor (KNR)

The KNR regressor, as shown in Section 8.1, is very similar to the KNN algorithm and hence we expect performances similar to the classifier one.

10.2.1 Rosenbrock Test

Starting from a grid 6×6 we perform the first regression test on a Rosenbrock test with a value of K equal to 3 as shown below:

```
<ROM name="ROM KNR" subType="SciKitLearn">
   <SKLtype>neighbors|KNeighborsRegressor</SKLtype>
   <Features>x1,x2</Features>
   <Target>y1</Target>
   <n neighbors>3</n neighbors>
   <algorithm>kd tree</algorithm>
</ROM>
```

RAVEN Input block 15 – KNR definition for the Rosenbrock function test

The reconstruction of the Rosenbrock function was very poor as shown in Figure 67. The plot of the relative error is shown in Figure 68. Such big errors are due to the fact that KNR is not actually based on a regression or interpolation kernel per se, but it is just averaging the value of the K nearest neighbors.

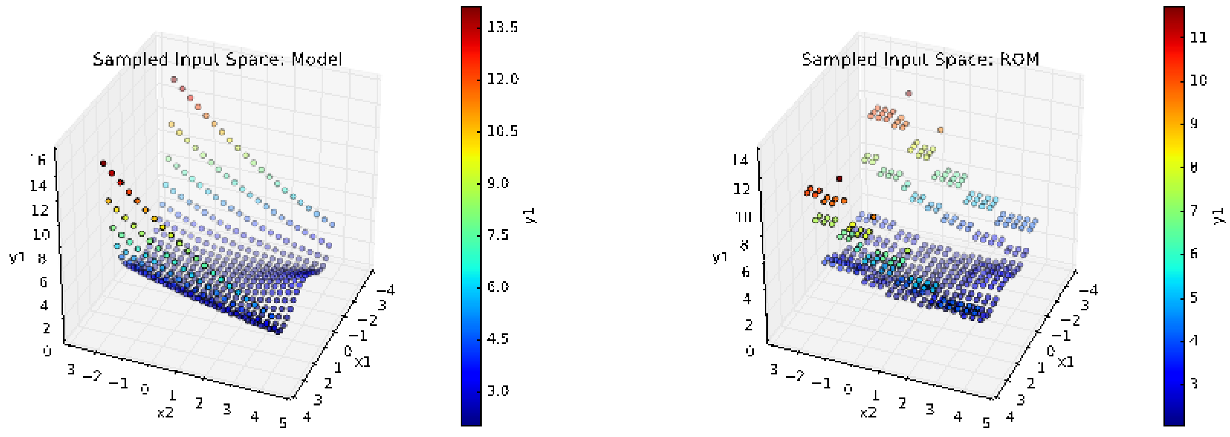


Figure 67 – KNR: Comparison of the response function for the Rosenbrock test case

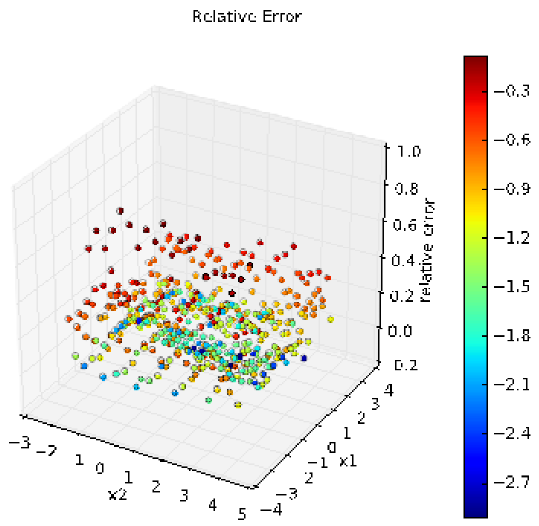


Figure 68 – KNR: Relative error for the Rosenbrock test case

10.2.2 Paraboloid test

We performed the second test on the Paraboloid test case in order to evaluate how the regression error propagates in the UQ/SA phase. Results are shown in Figure 69 Figure 70 and summarized in Table 9. Note even though the regression error is high, the propagation of uncertainties of the two uncertain input parameters smooth these error. However note how the distribution of *y1* in Figure 70 is fairly different from the theoretical one.

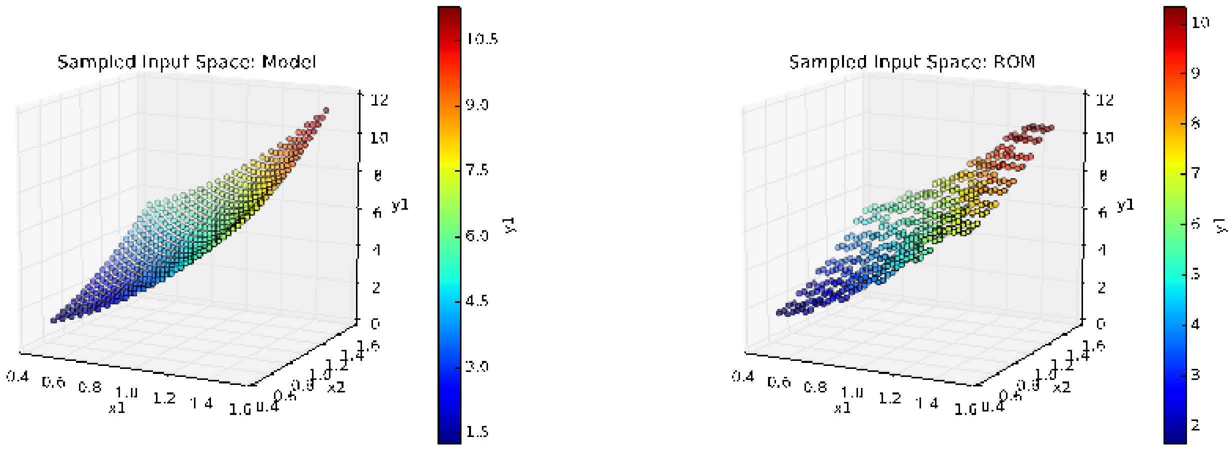


Figure 69 – KNR: Comparison of the response function for the paraboloid loop test case

Table 9 – KNR: Comparison of the statistical moments, Pearson and sensitivity coefficients for *y*1 for the paraboloid test case

	Reference	ROM	Relative Error
mu	5.18212435	5.23077005	-0.009387212
sigma	1.38749472	1.40330500	-0.01139484
skewness	0.339384699	0.334100967	0.015568563
x1 sensitivity	5.99808493	5.97614374	0.003658033
x2 sensitivity	4.00024747	3.98495764	0.003822221
x1 Pearson	0.825704716	0.813415336	0.014883505
x2 Pearson	0.552788789	0.544472021	0.01504511

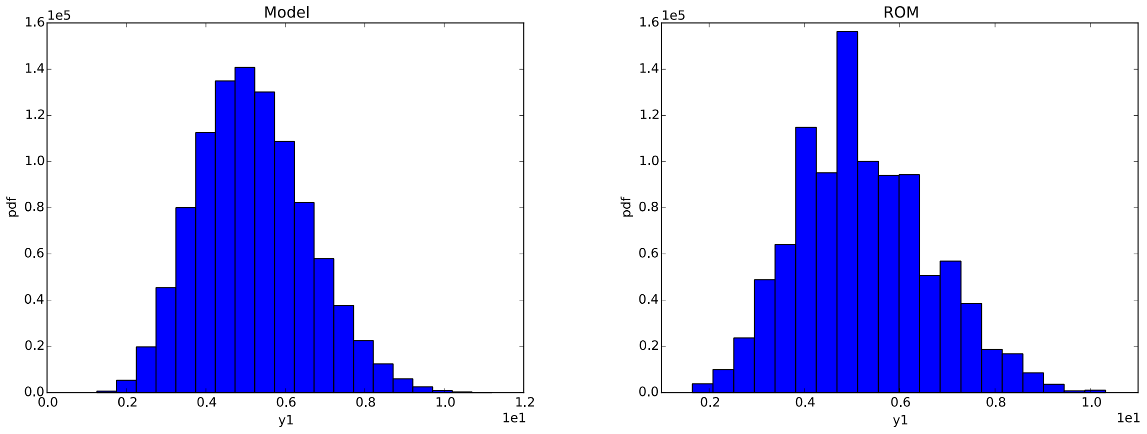


Figure 70 – KNR: Comparison of the pdfs of y1 for the paraboloid test case

10.2.3 PWR Natural Circulation Loop Test

We performed the same analysis shown in Section 10.1.2 but using the RELAP-7 PWR natural circulation loop and the KNR surrogate model as shown below (initial training grid was 6 × 6):

```
<ROM name="ROM_KNR" subType="SciKitLearn">
   <SKLtype>neighbors|KNeighborsRegressor</SKLtype>
   <Features>Materials|fuel-mat|k,Components|reactor|power</Features>
   <Target>PeakFuelTemperature</Target>
   <n_neighbors>3</n_neighbors>
   <algorithm>kd_tree</algorithm>
</ROM>
```

RAVEN Input block 16 – KNR definition for the Rosenbrock function test

Results are shown in Figure 71, Figure 72 and Figure 73 while they are summarized in Table 10. The same considerations explained in Section 10.1.3 are valid here.

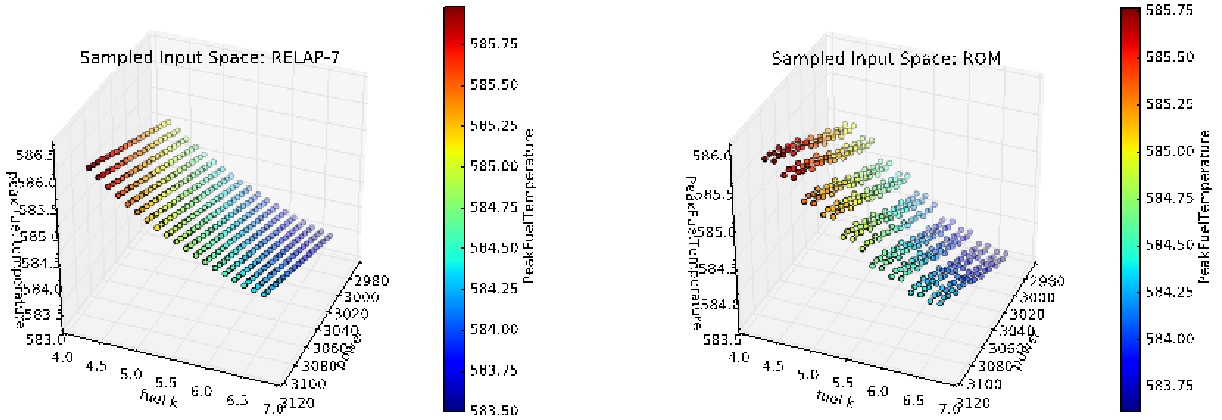


Figure 71 – KNR: Comparison of the response function for the PWR natural circulation test case

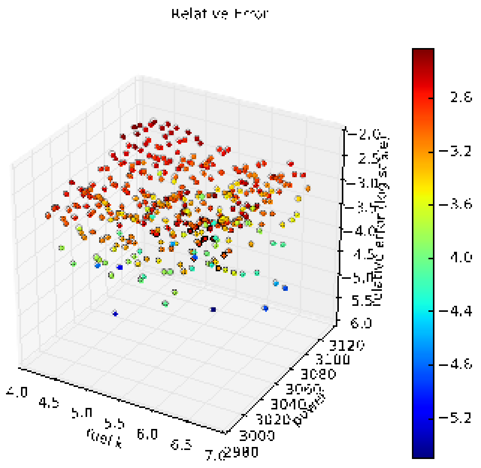


Figure 72 – KNR: max relative error for the PWR natural circulation loop test case

Table 10 - KNR: Comparison of the statistical moments, Pearson and sensitivity coefficients for y1 for the PWR natural circulation test case

	RELAP-7	ROM	Relative Error
mu	5.84550604 E+02	5.84557948 E+02	-1.25635E-05
sigma	3.46907782 E-01	3.51373560 E-01	-0.012873098
skewness	3.44692439 E-01	3.45760677 E-01	-0.003099105
K sensitivity	-6.66368173 E-01	-6.64299955 E-01	0.003103717
P sensitivity	7.23160579 E-03	7.22390622 E-03	0.001064711
K pearson	-9.05607661 E-01	-8.91323155 E-01	0.015773394
P pearson	-3.97276934 E-04	-3.97276934 E-04	0

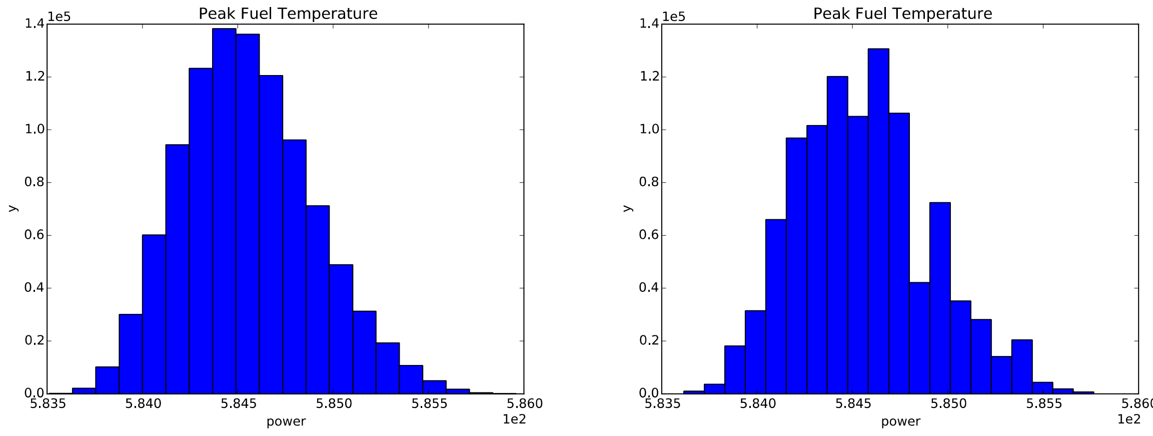


Figure 73 – KNR: Comparison of the pdfs of peak fuel temperature for the PWR natural circulation test case

10.2.4 Convergence Study on KNR

Regarding the convergence study of KNR we performed a series of test to perform the performances of KNR algorithms by changing the initial training grid and the value of the parameter *K*.

Table 11 – KNR: Max relative error as function of K and grid size for the Rosenbrock test case

		Grid				
		3	5	7	9	11
K	3	1.42067684	0.83274725	0.29376348	0.32445469	0.18495505
	4	1.91869853	1.19110615	0.53416133	0.51789447	0.25364263
	5	1.63524659	0.92473063	0.58646759	0.42982842	0.31561579
	6	2.00251335	1.30171444	0.83733852	0.55191956	0.34646707
	8	2.44297832	1.12511214	0.72334650	0.48676474	0.44874565
	11	2.15429607	1.09436904	0.73316160	0.56736624	0.42097214

We expect the max relative error in the reconstruction of the response surface decreases when the initial grid size increases. In addition, we also expect that by increasing the value of K such error is increasing since more neighbors are considered in each prediction. Obtained results confirm our prediction and they are shown in Figure 74 and Figure 75 and they are summarized in Table 11.

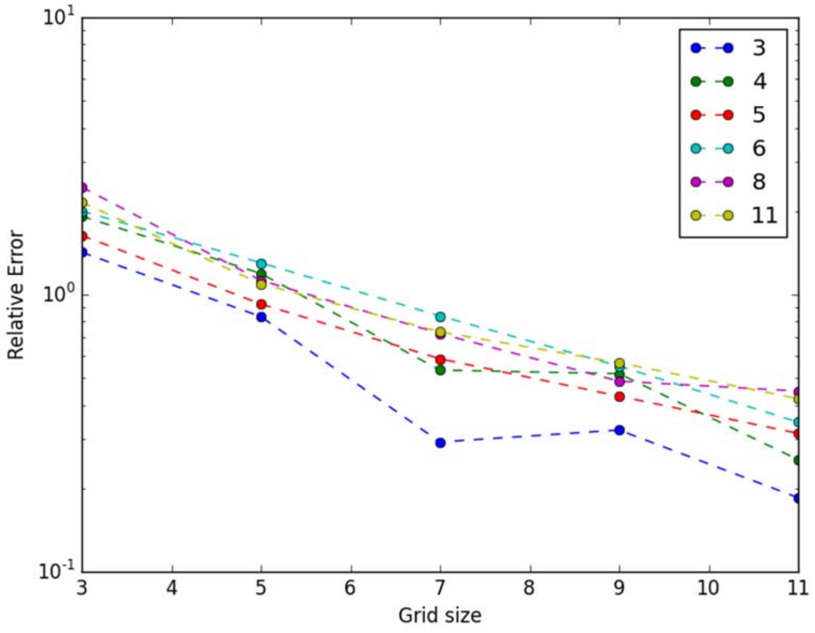


Figure 74 – KNR: plot of the max relative error as function of K and grid size for the Rosenbrock test case

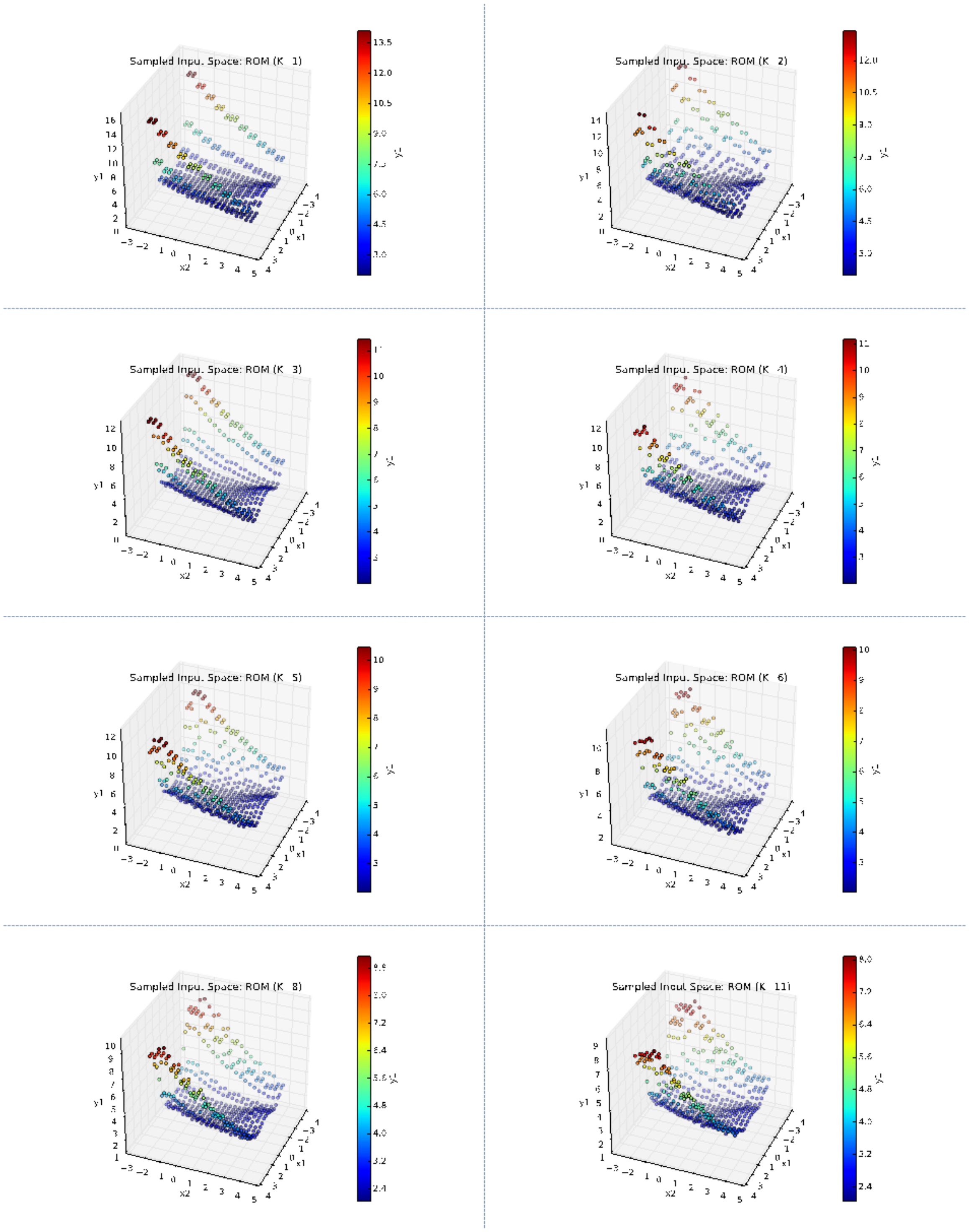


Figure 75 – KNR: Response function plots as function of K

10.3 Support Vector Regressor (SVR)

10.3.1 Rosenbrock Test

Starting from a grid 6 × 6 we perform the first regression test on a Rosenbrock test with a value of *Gamma* and *C* as shown below:

```
<ROM name="ROM_SVR" subType="SciKitLearn">
   <SKLtype>svm|SVR</SKLtype>
   <Features>x1,x2</Features>
   <Target>y1</Target>
   <kernel>rbf</kernel>
   <C>100000</C>
   <gamma>0.1</gamma>
</ROM>
```

RAVEN Input block 17 – SVR definition for the Rosenbrock function test

Even though the reconstruction of the response surface looks accurate as shown in Figure 76, the maximum relative error was 0.083497 (see Figure 77).

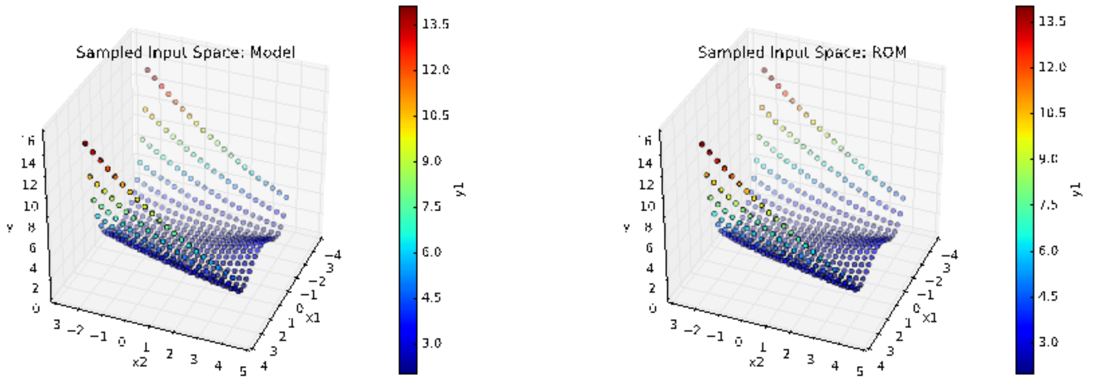


Figure 76 – SVR: Comparison of the response function for the Rosenbrock test case

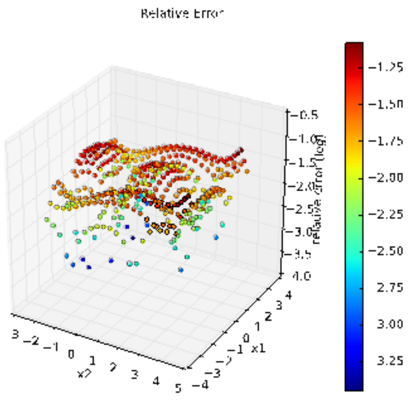


Figure 77 – SVR: Comparison of the response function for the Rosenbrock test case

10.3.2 Paraboloid test

We performed the second test on the Paraboloid test case in order to evaluate how the regression error propagates in the UQ/SA phase. Results are shown in and summarized in Figure 78 and Figure 79 and they are summarized in Table 12.

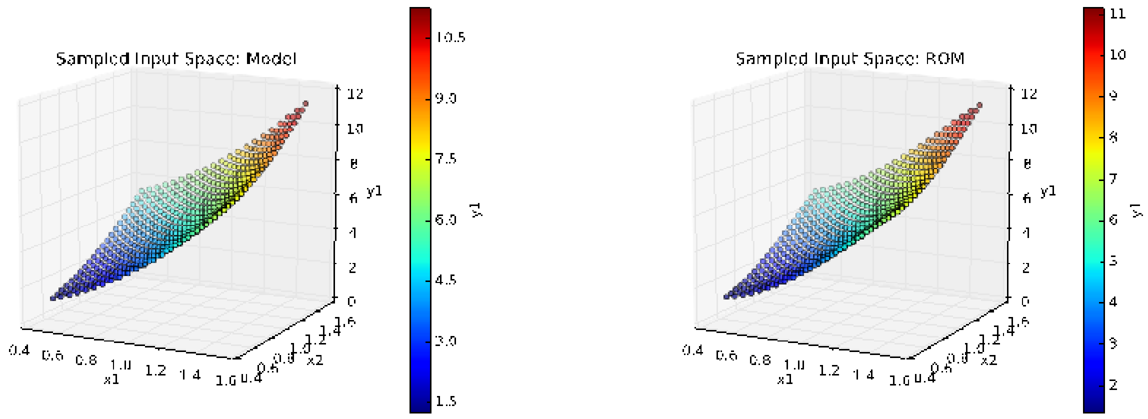


Figure 78 – SVR: Comparison of the response function for the paraboloid test case

Table 12 – SVR: Comparison of the statistical moments, Pearson and sensitivity coefficients for y1 for the paraboloid test case

	Reference	ROM	Relative Error
mu	5.18212435	5.14685656	0.006805663
sigma	1.38749472	1.42465494	-0.026782242
skewness	0.339384699	0.360452670	-0.062076962
x1 sensitivity	5.99808493	6.17922605	-0.030199826
x2 sensitivity	4.00024747	4.05308554	-0.0132087
x1 Pearson	0.825704716	0828433888	-0.003305264
x2 Pearson	0.552788789	0545509840	0.013167686

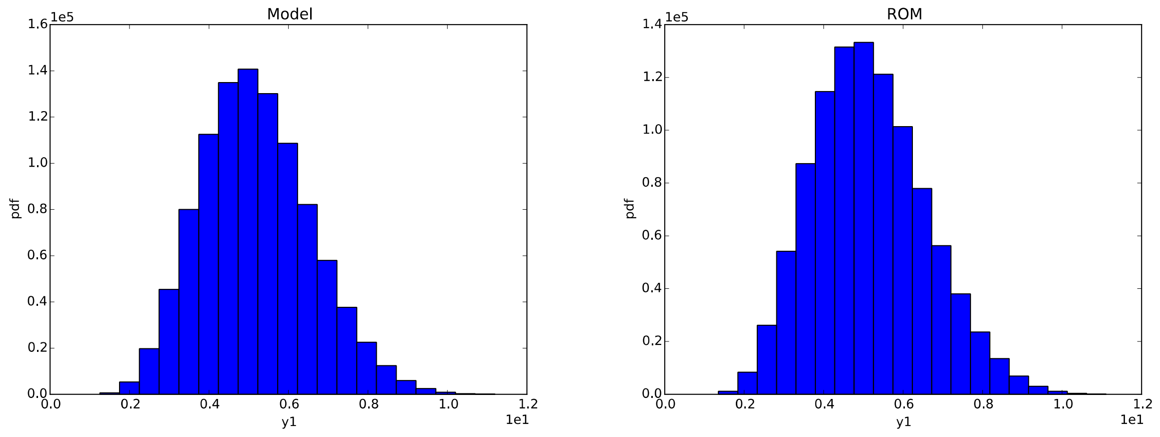


Figure 79 – SVR: Comparison of the pdfs of y1 for the paraboloid test case

10.3.3 PWR Natural Circulation Loop Test

We performed the same analysis shown in Section 10.1.3 but using the RELAP-7 PWR natural circulation loop and the SVR surrogate model as shown below (initial training grid was 6 × 6):

```
<ROM name="ROM SVR" subType="SciKitLearn">
   <SKLtype>svm|SVR</SKLtype>
   <Features>Materials|fuel-mat|k,Components|reactor|power</Features>
   <Target>PeakFuelTemperature</Target>
   <kernel>rbf</kernel>
   <C>1000000</C>
   <gamma>0.1</gamma>
</ROM>
```

RAVEN Input block 18 – SVR definition for the PWR Natural Circulation Loop test

The maximum relative error was max 0.0036869 as shown in Figure 80, Figure 81 and Figure 82 while they are summarized in Table 13.

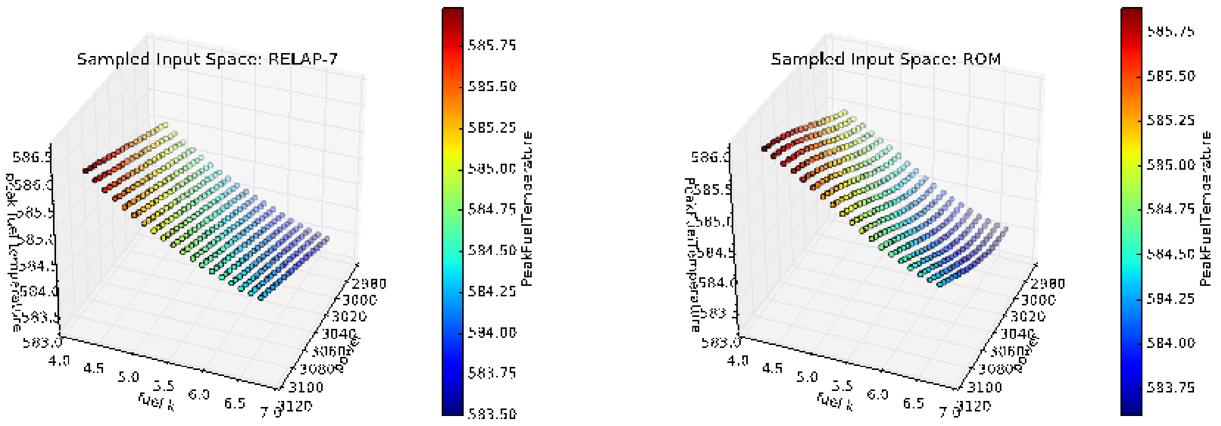


Figure 80 – SVR: Comparison of the response function for the PWR natural circulation loop test case

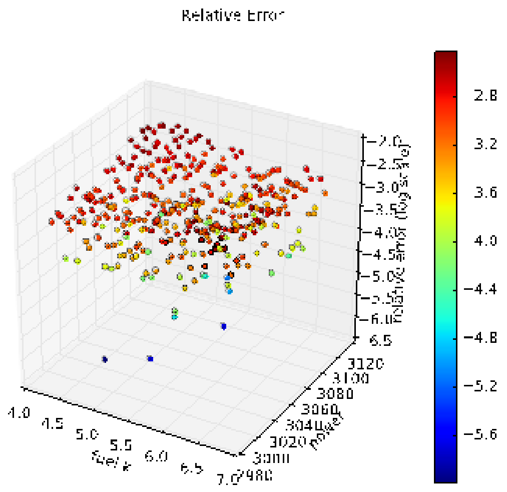
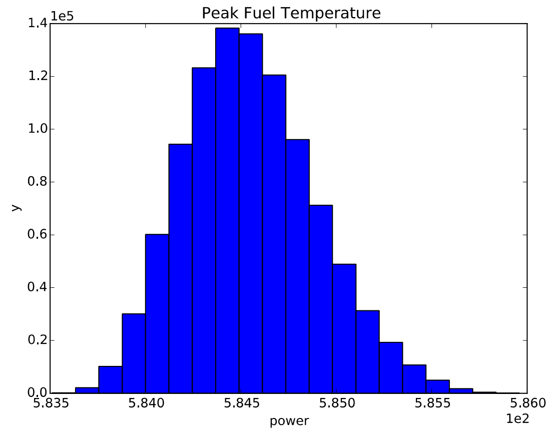


Figure 81 – SVR: max relative error for the natural curculation loop test case

Table 13 – SVR: Comparison of the statistical moments, Pearson and sensitivity coefficients for y1 for the PWR natural cicrulation loop test case

	RELAP-7	ROM	Relative Error
mu	5.84550604 E+02	5.84555067 E+02	-7.63492E-06
sigma	3.46907782 E-01	3.89928838 E-01	-0.124012946
skewness	3.44692439 E-01	2.38731506 E-01	0.307407187
K sensitivity	-6.66368173 E-01	-7.48461405 E-01	-0.123195007
P sensitivity	7.23160579 E-03	8.23918588 E-03	-0.139330063
K Pearson	-9.05607661 E-01	-9.04950999 E-01	0.000725106
P Pearson	-3.97276934 E-04	-3.97276934 E-04	0



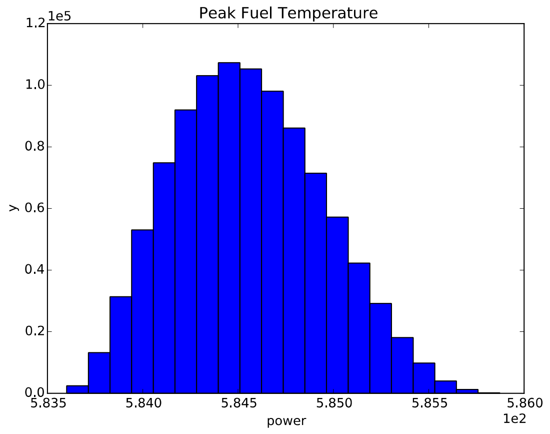


Figure 82 – SVR: Comparison of the pdfs of peak fuel temperature for the PWR natural circulation loop test case

10.3.4 Convergence study on SVR

As performed for the SVM classifier in Section 9.1.4 we have performed a series of convergence studies by changing the input parameters *Gamma* and *C* and by changing the initial training grid size. Results are shown in Figure 83 and they are summarized in Table 14.

Table 14 – SVR: Max relative error for the Rosenbrock test case as function of Gamma and C (initial training grid is 6x6)

		Gamma			
		0.01	0.1	1.0	10
C	1E4	0.911	0.0836	0.2707	0.5973
	1E5	0.8826	0.0835	0.2707	0.5973
	1E6	0.8627	0.0835	0.2707	0.5973
	1E7	0.81345	0.0835	0.2707	0.5973

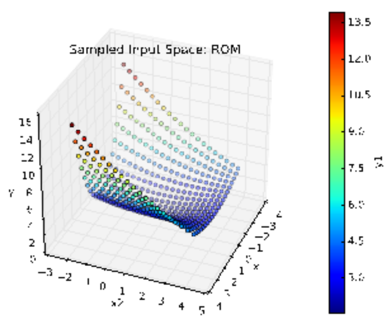
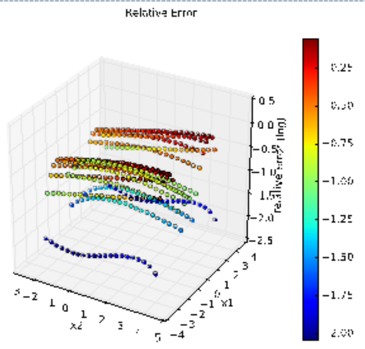
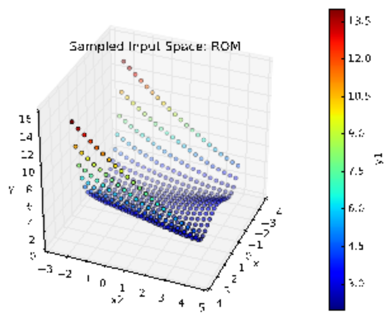
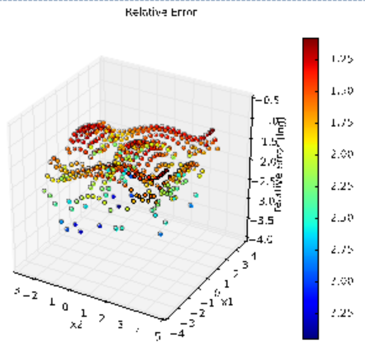
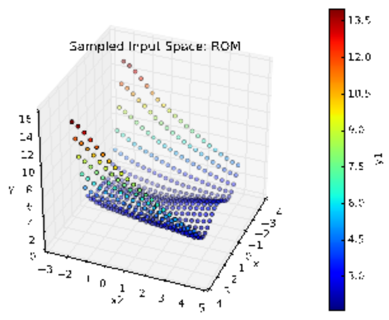
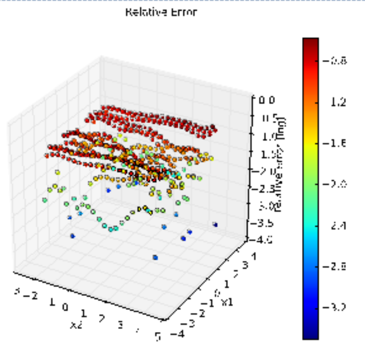
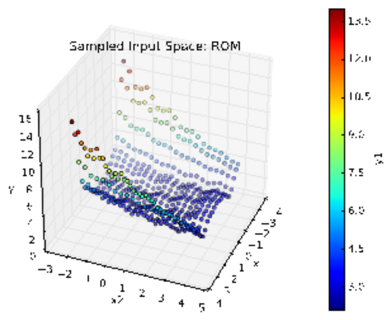
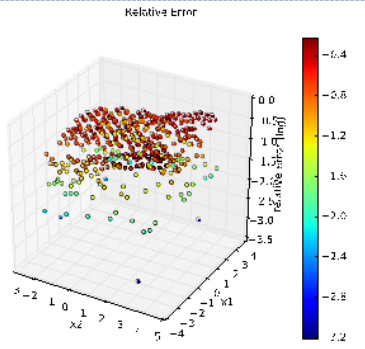
Gamma	Response Surface	Relative Error
0.01		
0.1		
1.0		
10.0		

Figure 83 – SVR: plots of the response function and relative error as function of gamma (C=1.0E5 and 5x5 grid)

We also evaluate the convergence of the algorithm when the initial grid size is changed. The rational is that by increasing the training sample size, the max relative error decreases. Results are summarized in Table 15 and they are plotted in Figure 84 by using the SVR surrogate model as indicated below:

```
<ROM name="ROM_SVR" subType="SciKitLearn">
   <SKLtype>svm|SVR</SKLtype>
   <Features>x1,x2</Features>
   <Target>y1</Target>
   <kernel>rbf</kernel>
   <C>100000</C>
   <gamma>0.1</gamma>
</ROM>
```

RAVEN Input block 19 – SVR definition for the convergence study

Table 15 – SVR: max relative error as function of the grid size for the Rosenbrock test case

Grid size	Relative error (%)
3	0.779250440154
5	0.0834973073469
7	0.0566070865042
9	0.0539084403364
11	0.0507307124977
14	0.0493425293902

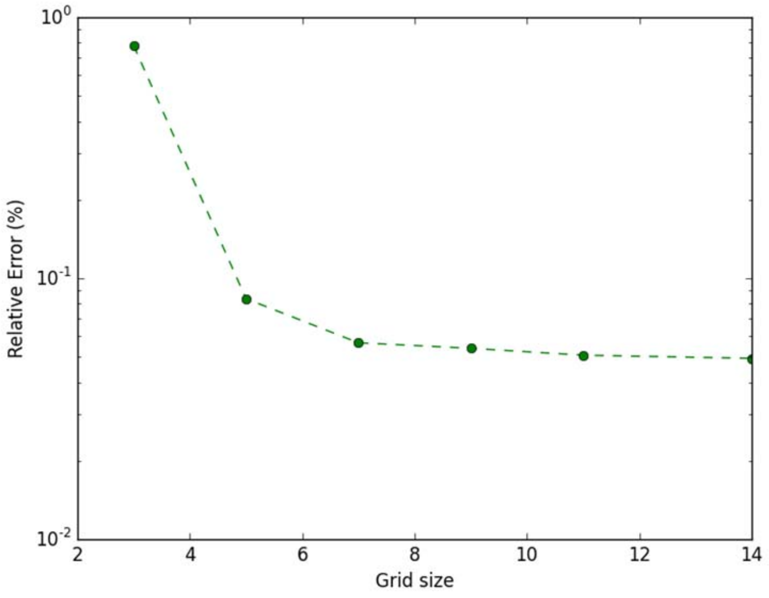


Figure 84 – SVR: Plot of the max relative error as function of the grid size

10.4 Sheppard Interpolation Method

In this section we tested the Sheppard interpolation method, also known as inverse distance weighting interpolation. This is the only method available in CROW that we tested.

10.4.1 Rosenbrock Test

Starting from a grid 6 × 6 we perform the first regression test on a Rosenbrock test with a value of *p* equal to 3 as shown below:

```
<ROM name="ROM_IW" subType="NDinvDistWeight">
   <Features>x1,x2</Features>
   <Target>y1</Target>
   <p>3</p>
</ROM>
```

RAVEN Input block 20 – Sheppard Method definition for the Rosenbrock Test

Results are shown in Figure 85 and Figure 86. Analogously to the KNR regressor, the interpolation is done by considering not *K* but all training points weighted by the inverse of the distance. Even though these methods are flexible to model very discontinuous response surfaces, they are affected by heavy interpolation error.

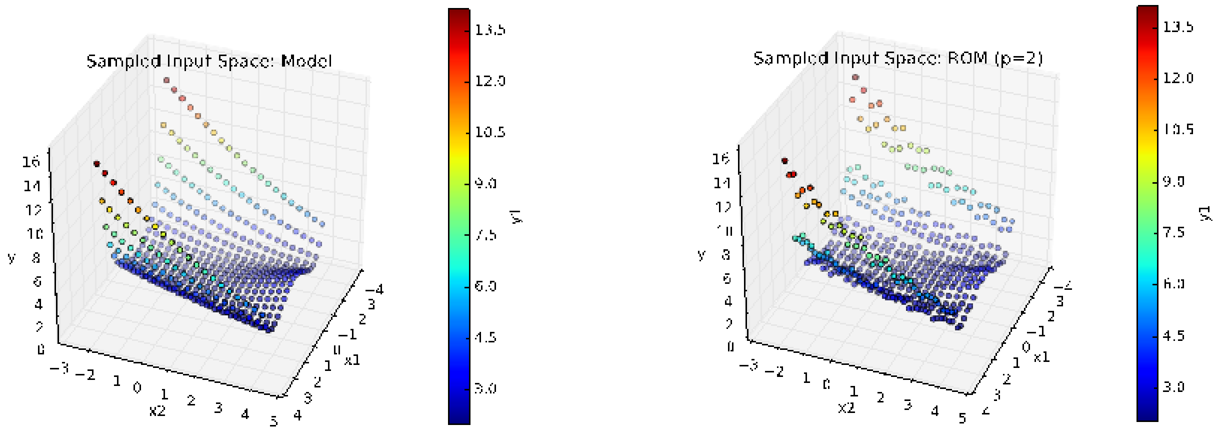


Figure 85 – Sheppard method: Comparison of the response function for the Rosenbrock test case

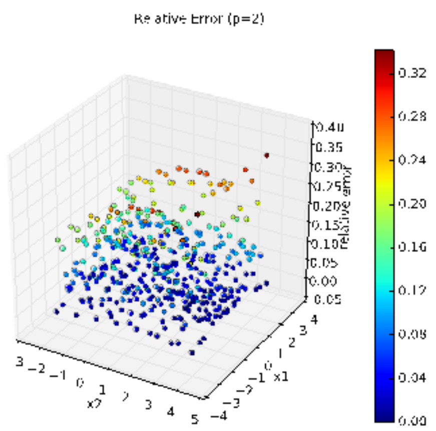


Figure 86 – Sheppard method: max relative error for the Rosenbrock test case

10.4.2 Paraboloid test

Starting from a grid 6 × 6 we perform the first regression test on the Paraboloid test again with a value of *p* equal to 3. Results are shown in Figure 87 and Figure 88 and they are summarized in Table 16.

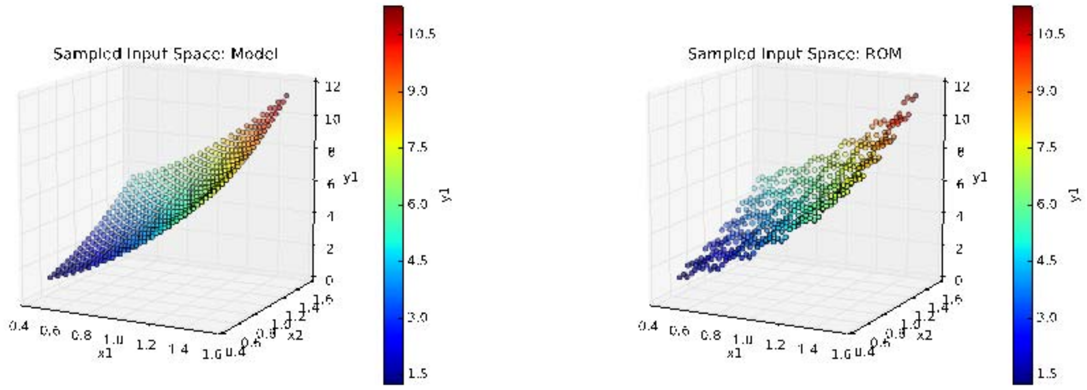


Figure 87 – Sheppard method: Comparison of the response function for the paraboloid test case

Table 16 – Sheppard method: Comparison of the statistical moments, Pearson and sensitivity coefficients for *y1* for the paraboloid test case

	Reference	ROM	Relative Error
mu	5.18212435	5.23847510	-0.010874064
sigma	1.38749472	1.34709937	0.029113877
skewness	0.339384699	0.327595986	0.034735547
x1 sensitivity	5.99808493	5.78141655	0.036122926
x2 sensitivity	4.00024747	3.85665964	0.035894737
x1 Pearson	0.825704716	0.819744038	0.007218898
x2 Pearson	0.552788789	0.548927597	0.006984932

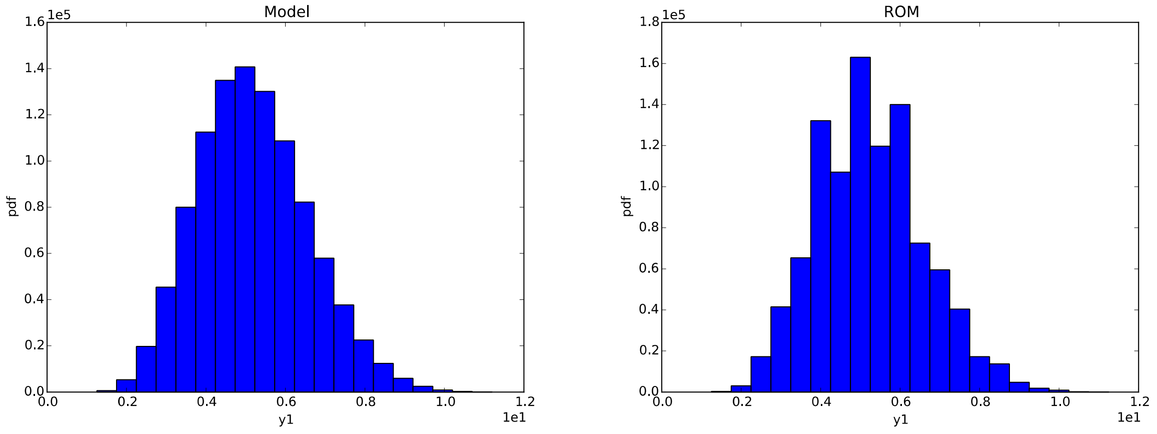


Figure 88 – Sheppard method: Comparison of the pdfs of y1 for the paraboloid test case

10.4.3 PWR Natural Circulation Loop Test

Starting from a grid 6x6 we perform the first regression test on the Paraboloid test again with a value of *p* equal to 3. Results are summarized in Table 17 and plotted in Figure 89, Figure 90 and Figure 91. The max relative error was in this case equal to 0.0037801. Again note how the evident errors in the reconstruction of the response surface.

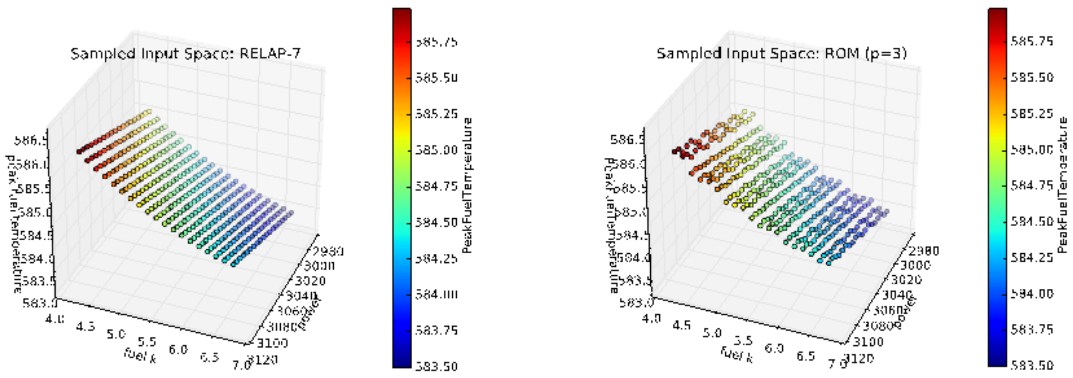


Figure 89 – Sheppard method: Comparison of the response function for the PWR natural circulation loop test case

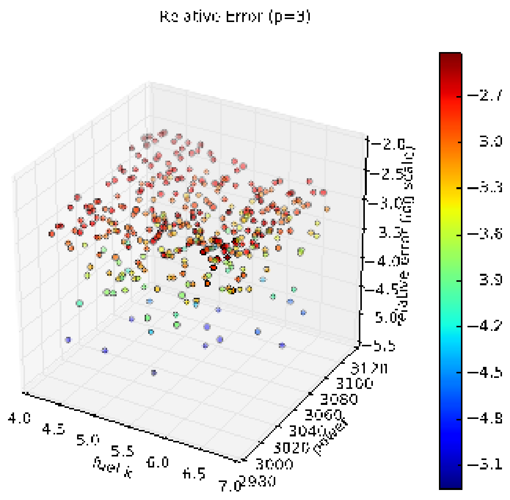


Figure 90 – Sheppard method: max relative error for the PWR natural circulation loop

Table 17 – Sheppard method: Comparison of the statistical moments, Pearson and sensitivity coefficients for *y1* for the PWR natuaral circulation loop test case

	RELAP-7	ROM	Relative Error
mu	5.84550604 E+02	5.84559269 E+02	-1.48234 E-05
sigma	3.46907782 E-01	3.37515077 E-01	0.02707551
skewness	3.44692439 E-01	3.44898676 E-01	-0.000598322
K sensitivity	-6.66368173 E-01	-6.43454798 E-01	0.034385458
P sensitivity	7.23160579 E-03	6.98035452 E-03	0.034743496
K pearson	-9.05607661 E-01	-8.98803428 E-01	0.007513445
P pearson	-3.97276934 E-04	-3.97276934 E-04	0

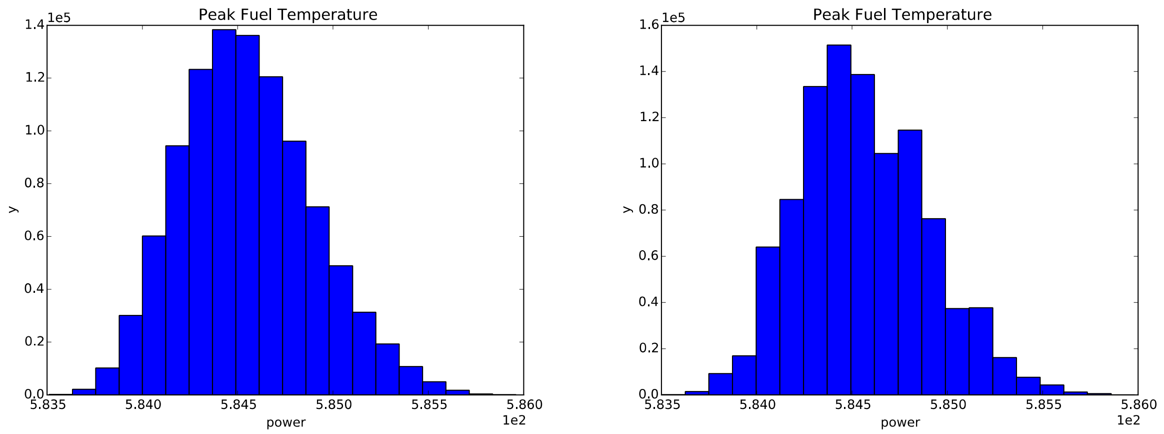


Figure 91 – Sheppard method: Comparison of the pdfs of peak fuel temperature for the PWR natural circulation loop test case

10.4.4 Convergence study on Sheppard interpolation method

As performed for the SVM classifier in Section 10.1.4 we have performed a series of convergence studies by changing the input parameter *p* and by changing the initial training grid size. We used the Rosenbrock function as test model. Obtained results are summarized in Table 18. Again note how the max relative error is fairly even if the grid size is increased.

Table 18 – Sheppard Method: max relative error as function of the grid size and p

		p			
		1	2	3	4
Grid size	3	1.0487	1.0614	1.0694	1.0732
	5	0.4419	0.45456	0.45819	0.45917
	7	0.34776	0.34160	0.3366	0.33380
	9	0.2329	0.2246	0.2242	0.2256
	11	0.16079	0.13850	0.1377	0.1377
	14	0.14035	0.13059	0.13076	0.132118

10.5 Generalized Polynomial Chaos expansion (gPC)

The last regressor to be tested has been internally developed by the RAVEN team: gPC. The main difference of this type of surrogate models is that its creation coupled with a specific sampler available in RAVEN: Sparse Grid Collocation sampler. This is due to the fact that the number and the location of the training samples for the surrogate model that are generated by the sampler depend on choice iof the internal parameters of the surrogate model. The definition of this sampler and its surrogate mdoel is shown below:

```
<Samplers>
   <SparseGridCollocation name="SG" parallel="0">
   <variable name="x1">
      <distribution>normal</distribution>
   </variable>
   <variable name="x2">
      <distribution>normal</distribution>
   </variable>
   <ROM class="Models" type="ROM">GP_rom</ROM>
   </SparseGridCollocation>
</Samplers>
<Models>
   <ROM name="GP_rom" subType="GaussPolynomialRom">
      <Features>x1,x2</Features>
      <Target>y1</Target>
      <IndexSet>TotalDegree</IndexSet>
      <PolynomialOrder>4</PolynomialOrder>
      <Interpolation quad="Hermite" poly="Hermite" weight="1">x1</Interpolation>
      <Interpolation quad="Hermite" poly="Hermite" weight="1">x2</Interpolation>
   </ROM>
</Models>
```

RAVEN Input block 21: gPC definition for the Rosenbrock Test

10.5.1 Rosenbrock Test

The first testing of this surrogate model was the Rosenbrock function. Results are shown in Figure 92 and Figure 93. We employed a fourth order polynomial and the max relative error was extremely low: 4 order 1.33545 E-10.

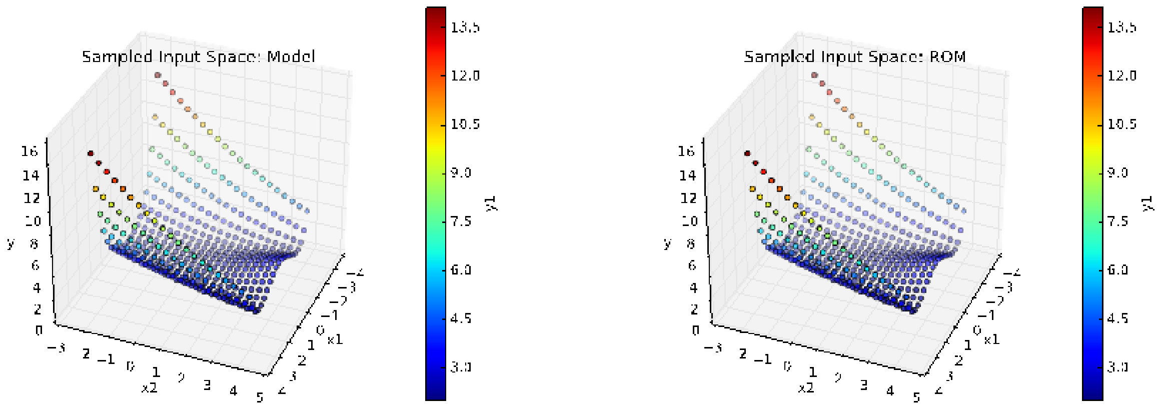


Figure 92 – gPC: Comparison of the response function for the Rosenbrock test case

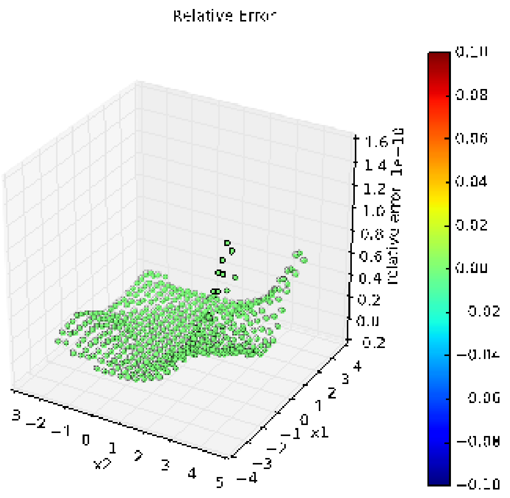
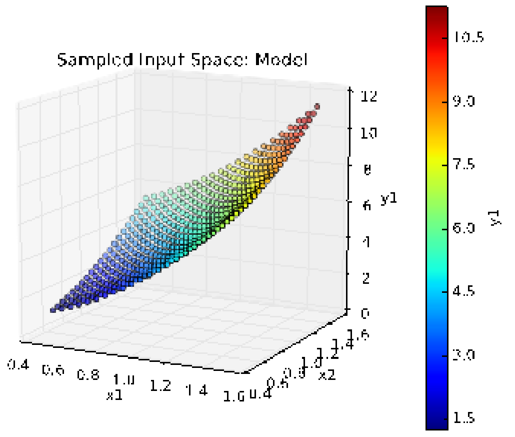


Figure 93 – gPC: max relative error for the Rosenbrock test case

10.5.2 Paraboloid test

We then tested the gPC surrogate model for the analytical UQ/SA test. Results are shown in Figure 94 and Figure 95 and they are shown in

Table 19. Remember that, if a comparison with the other surrogate models of this section is made, the training set for the gPC surrogate models is not a 6 × 6 grid; the actual number of samples and their location actually depends on the polynomial order. For this case we chose a 4th order polynomial which required 53 training points. Even though more training points were used (53 instead of 36) the obtained results were much better.



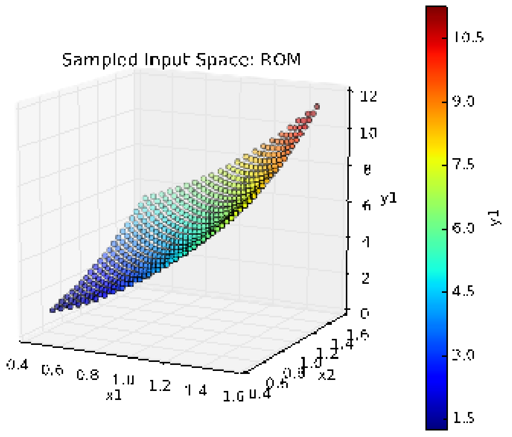


Figure 94 – gPC: Comparison of the response function for the paraboloid test case

Table 19 – gPC: Comparison of the statistical moments, Pearson and sensitivity coefficients for y1 for the paraboloid test case

	Reference	ROM	Relative Error
mu	5.18212435	5.18112884	0.000192105
sigma	1.38749472	1.38757635	-5.88327 E-05
skewness	0.339384699	0.341472156	-0.00615071
x1 sensitivity	5.99808493	5.99837735	-4.87522 E-05
x2 sensitivity	4.00024747	3.99844840	0.00044974
x1 Pearson	0.825704716	0.825956678	-0.000305148
x2 Pearson	0.552788789	0.551648536	0.002062728

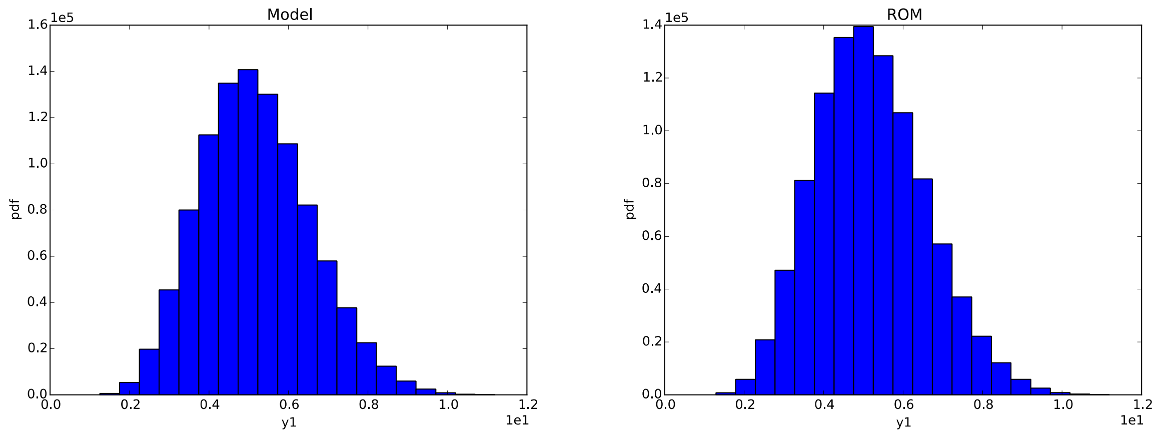


Figure 95 – gPC: Comparison of the pdfs of y1 for the paraboloid test case

10.5.3 PWR Natural Circulation Loop Test

We then repeated the UQ/SA analysis suing the RELAP-7 PWR natural circulation loop. Results are summarized in Table 20 and they are shown in Figure 96 and Figure 97. Again remember that, if a comparison with the other surrogate models of this section is made, the training set for the gPC surrogate models is not a 6×6 grid; the actual number of points and their location actually depends on the polynomial order. For this case we chose a 4th order polynomial which required 53 training points (instead of 36).

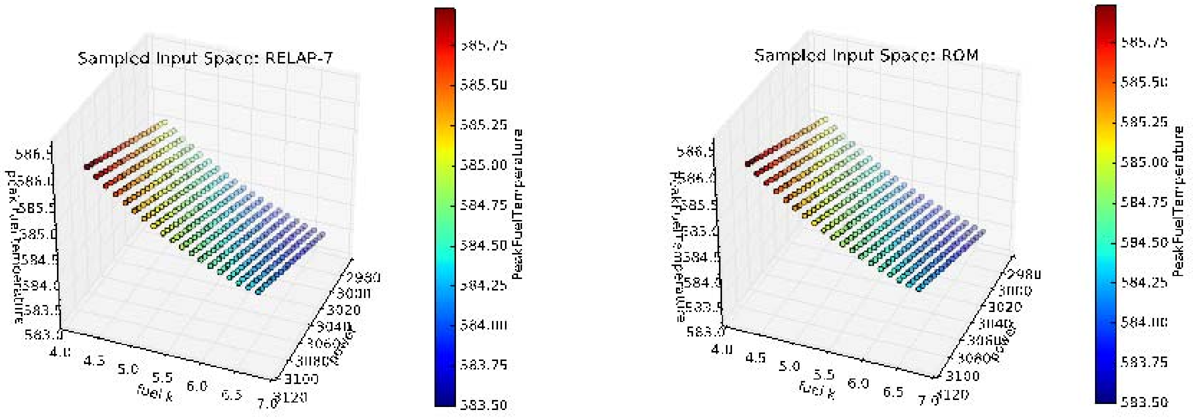


Figure 96 – gPC: Comparison of the response function for the PWR natural circulation loop test case

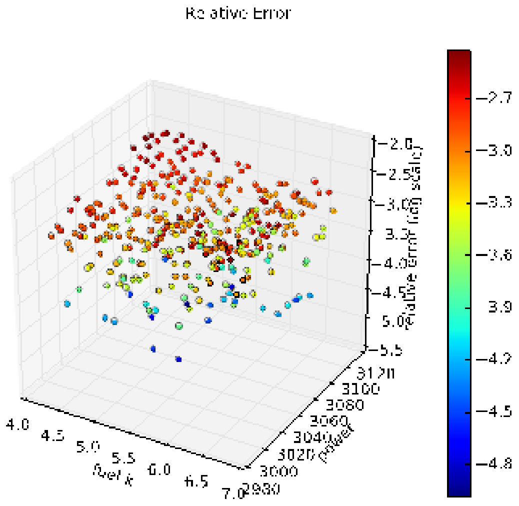


Figure 97 – gPC: max relative error for the PWR natural circulation loop test case

Table 20 – GPM: Comparison of the statistical moments, Pearson and sensitivity coefficients for y1 for the PWR natural circulation loop test case

	RELAP-7	ROM	Relative Error
mu	5.84550604 E+02	5.84550676 E+02	-1.23172 E-07
sigma	3.46907782 E-01	3.46943545 E-01	-0.000103091
skewness	3.44692439 E-01	3.44610341 E-01	0.000238178
K sensitivity	-6.66368173 E-01	-6.66135712 E-01	0.000348848
P sensitivity	7.23160579 E-03	7.23124314 E-03	5.01479E-05
K pearson	-9.05607661 E-01	-9.05683162 E-01	-8.33705E-05
P pearson	-3.97276934 E-04	-3.97276934 E-04	0

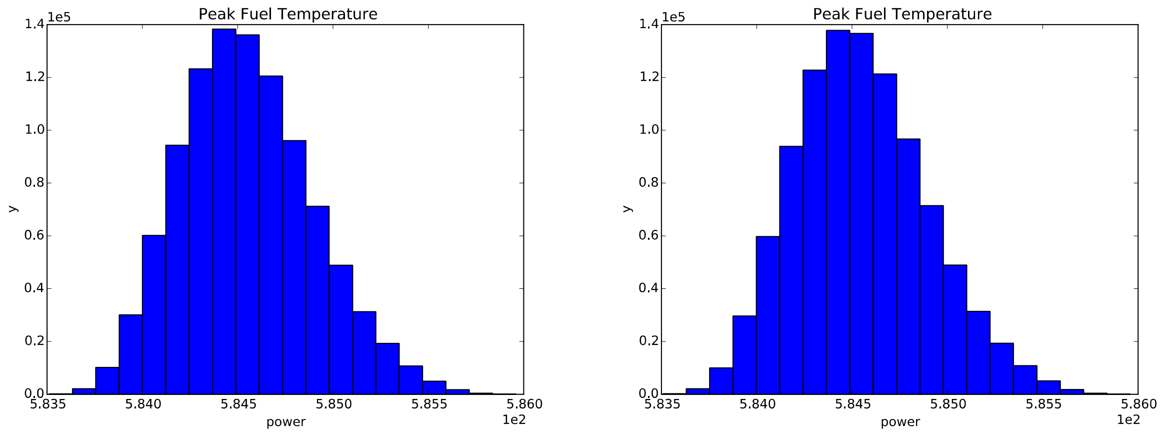


Figure 98 – gPC: Comparison of the pdfs of peak fuel temperature for the PWR natural circulation loop test case

10.5.4 Convergence study on GPC

For the scope of this report we focused on the evaluation the performances of the gPC surrogate models by increasing the polynomial order. Results are shown in Figure 99 and Figure 100 and they are summarized in Table 21. Note the discontinuous behavior of such error when the polynomial order moves form 3 to 4. It appears that the minimum polynomial order to capture the shape of the response function is 4.

Figure 102 shows the location of the samples generated by the associated sampler as function of the polynomial order. Recall that the training of the gPC surrogate model is tightly coupled to the generation of the training data; i.e., the training data was not identical to the one used by the other surrogate models presented in this section (i.e., a 6 × 6 grid.)

Table 21 – gPC: max relative error for the Rosenbrock test case as function of the polynomial order

Pol. Order	Max Relative error
2	1.79859462747
3	3.34216332644
4	1.33544718371 E-10
5	1.16275323726 E-10
6	3.94086123497 E-08

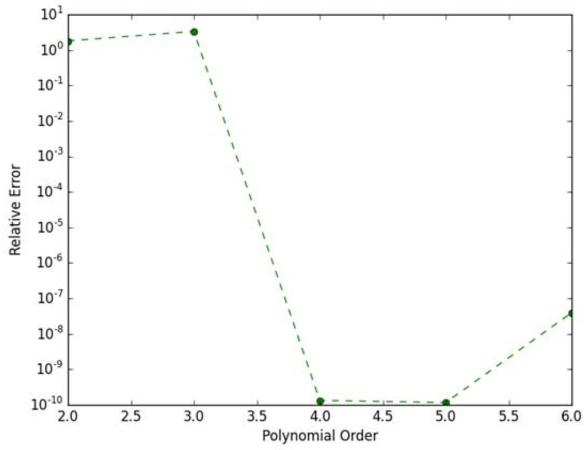


Figure 99 – gPC: plot of the max relative error for the Rosenbrock test case as function of the polynomial order

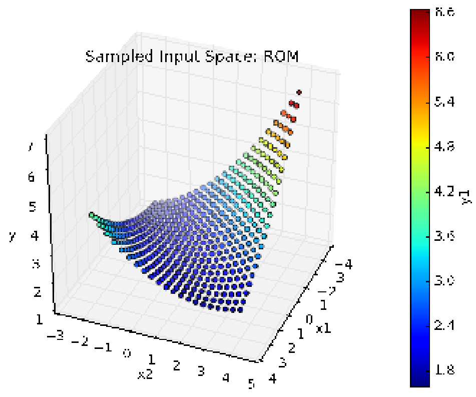
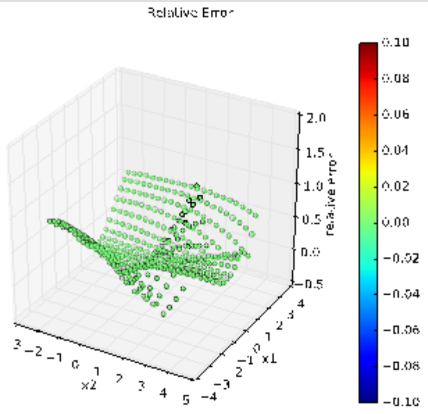
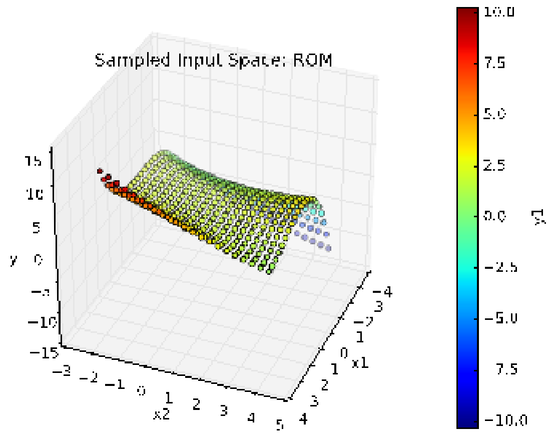
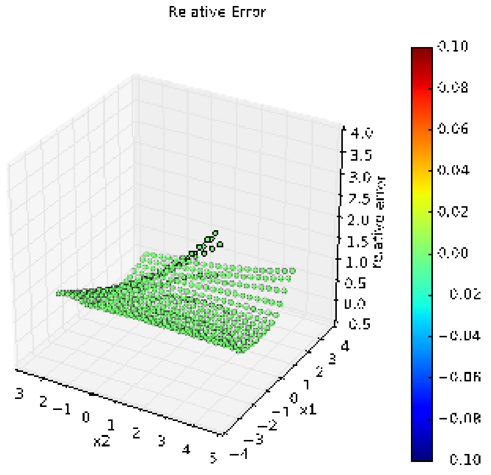
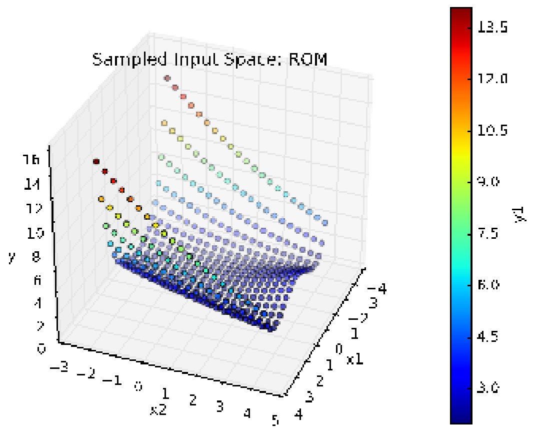
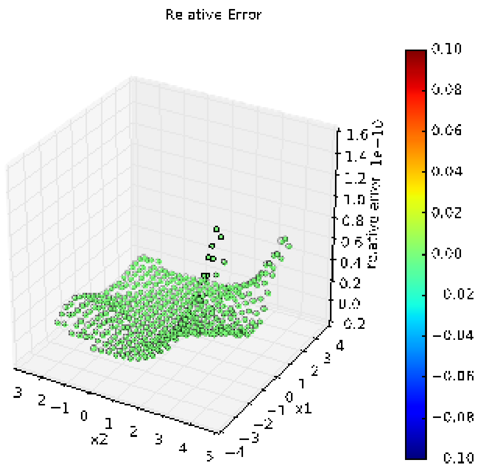
Pol. Ord.	Response surface	Error
2		
3		
4		

Figure 100 – gPC: plot of the response surface and the max relative error for the Rosenbrock test case as function of the polynomial order

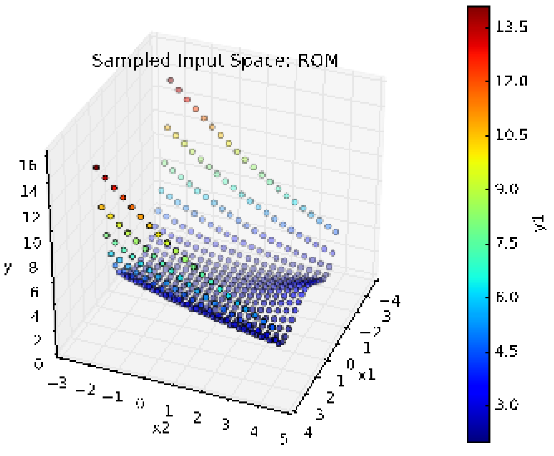
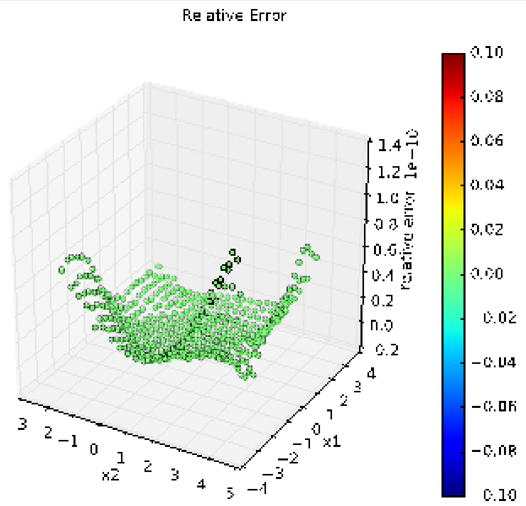
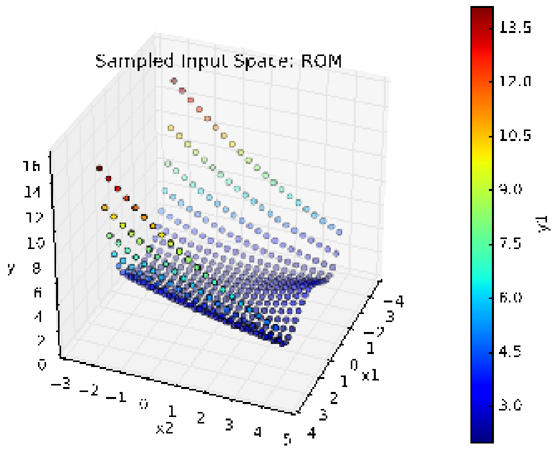
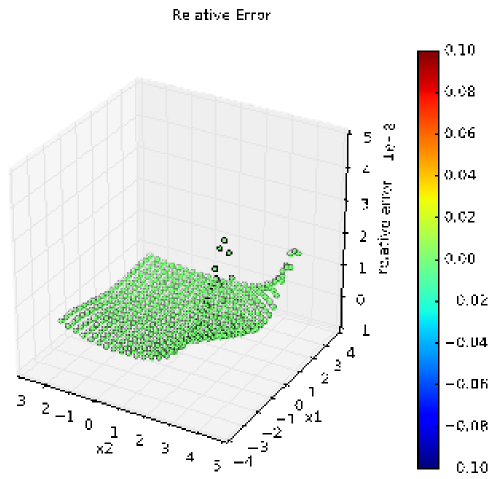
Pol. Ord.	Response surface	Error
5		
6		

Figure 101 – gPC: plot of the response surface and the max relative error for the Rosenbrock test case as function of the polynomial order

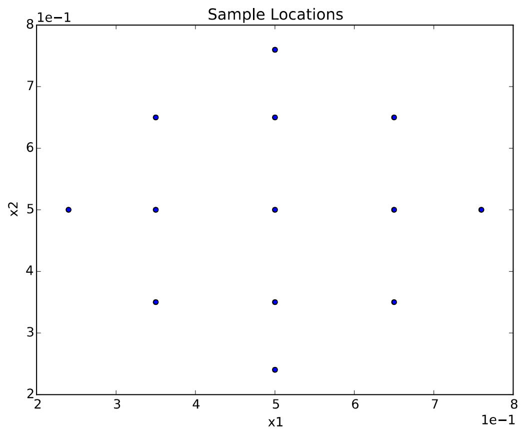
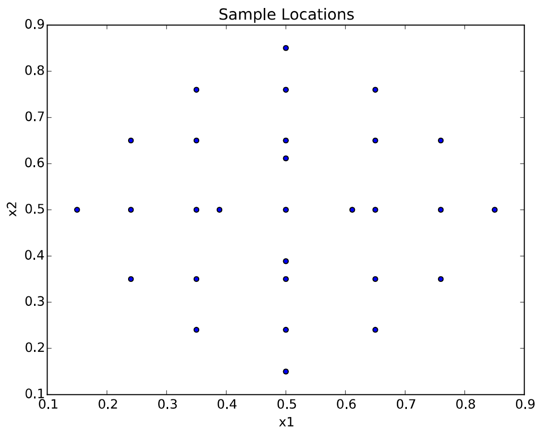
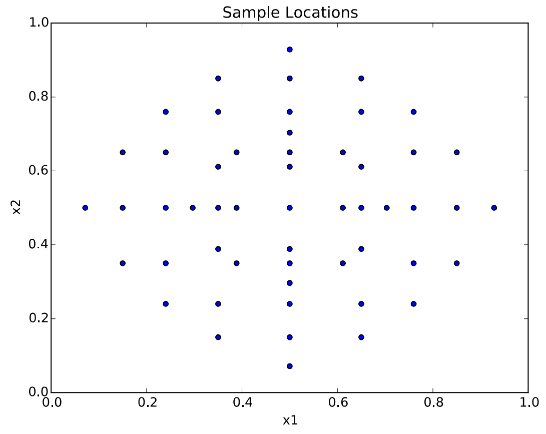
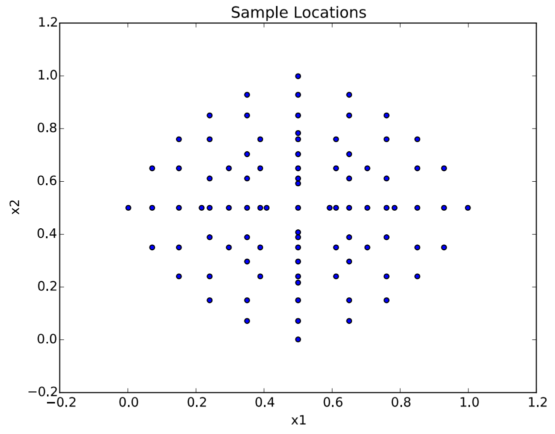
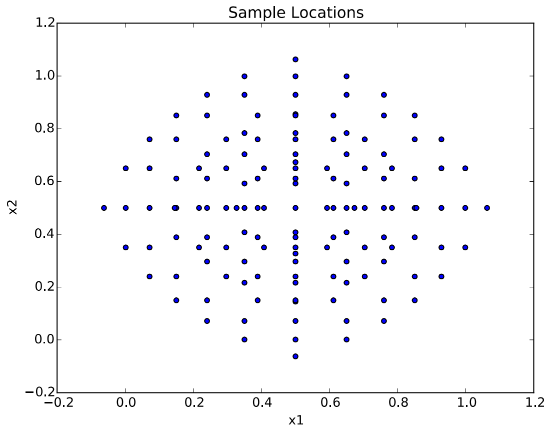
polynomial order = 2	**polynomial order = 3**
	
polynomial order = 4	**polynomial order = 5**
	
polynomial order = 6	
	

Figure 102 – gPC: plot of the sample locations for the Rosenbrock test case as function of the polynomial order

11. SUMMARY AND CONCLUSIONS

In this report we have given an overview of the Reduced Order Modeling capabilities available in the RISMC toolkit and, in particular, in the RAVEN statistical framework. We have shown how Reduced Order Modeling can be applied in a typical RISMC analysis: from the generation to the analysis and to visualization of data. We have shown how surrogate models can be used as a substitute for actual code to speed up the statistical analysis required by the RISMC approach. We have indicated how it is possible to reduce the computational cost of such statistical analyses by smartly sampling the input space.

The most important application of reduced order modeling techniques focuses on propagation of uncertainties and sensitivity analysis types of applications. In this respect we employed a few thermo-hydraulic models of RELAP-7 and we showed how this process can be performed by using the RAVEN code. Also, we have shown how reduced order modeling techniques can be also employed for data mining types of applications to visualize high dimensional data and extract useful information from large amounts of data.

We have, in particular, focused on surrogate models, both classifiers and regressors. We have tested their performances on a set of analyses that are of interest in the RISMC approach. We used not only analytical tests but also tests that involved RELAP-7 systems analysis. The comparison allowed us to identify the pros and cons of each algorithm and identify the best surrogate model depending on the case under consideration.

This overview was not however exhaustive since additional surrogate models are available in RAVEN. The set of algorithms we chose is a representative set of methods that most likely will be used within the RISMC project.

REFERENCES

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

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

[3] D. Mandelli, C. Smith, A. Alfonsi, C. Rabiti, J. Cogliati,” Improved Sampling Algorithms in the Risk-Informed Safety Margin Characterization Toolkit”, Idaho National Laboratory technical report: INL/EXT-15-35933 (2015).

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

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

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

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

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

[9] D. Gaston, C. Newman, G. Hansen and D. Lebrun-Grandi, “MOOSE: A parallel computational framework for coupled systems of nonlinear equations,” *Nuclear Engineering Design*, **239**, pp. 1768-1778 (2009).

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

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

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

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

[14] A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, R. Kinoshita, and A. Naviglio, “RAVEN and dynamic probabilistic risk assessment: Software overview,” in Proceedings of European Safety and Reliability Conference (ESREL 2014), Wroklaw (Poland), 2014

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

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

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

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

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

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

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

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

[23] 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).

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

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

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

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

[28] S. T. Roweis and L. K. Saul, ``Nonlinear dimensionality reduction by locally linear embedding," Science, vol. 290, pp. 2323-2328 (2000).

[29] H. Zhao, L. Zou, H. Zhang, R. Martineau, "Numerical Verification of RELAP-7 Model For A Single Phase Natural Circulation Loop", *in Proceeding of American Nuclear Society (ANS)*, San Antonio (2015).

[30] RELAP5-3D Code Development Team, RELAP5-3D Code Manual. 2005.

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

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

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

[34] H. Carr, J. Snoeyink, and U. Axen, "Computing contour trees in all dimensions", in Proceedings 11th Annual ACM-SIAM Symposium on Discrete algorithms, 2000, pp. 918–926.

[35] V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas, "Robust on-line computation of reeb graphs: simplicity and speed", ACM Transactions on Graphics, vol. 26, no. 3, p. 58, 2007.

[36] S. Gerber, P.-T. Bremer, V. Pascucci, and R. T. Whitaker, "Visual exploration of high dimensional scalar functions", IEEE Transactions on Visualization and Computer Graphics, vol. 16, pp. 1271–1280 (2010).

[37] D. Maljovec, B. Wang, V. Pascucci, P.-T. Bremer, M. Pernice, D. Mandelli, and R. Nourgaliev, "Exploration of high-dimensional scalar function for nuclear reactor safety analysis and visualization," in *Proceeding of M&C2013 International Topical Meeting on Mathematics and Computation*, CD-ROM, American Nuclear Society, LaGrange Park, IL (2013).

[38] D. Maljovec, B. Wang, D. Mandelli, P.-T. Bremer and V. Pascucci, "Analyzing Dynamic Probabilistic Risk Assessment Data through Topology-Based Clustering," in *ANS PSA 2013 International Topical Meeting on Probabilistic Safety Assessment and Analysis*, Columbia, SC, on CD-ROM, American Nuclear Society, LaGrange Park (IL) (2013).

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

[40] R. Duda, P. Hart, and D. Stork, Pattern Classification. Wiley-Interscience Publication, 2000.

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

[42] D. Maljovec, A. Saha, P. Lindstrom, P.-T. Bremer, B. Wang, C. Correa, and V. Pascucci, “A comparative study of morse complex approximation using different neighborhood graphs.” Workshop on Topological Methods in Data Analysis and Visualization (accepted), 2013.

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

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

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

[46] Wiener N. (October 1938). "The Homogeneous Chaos". American Journal of Mathematics (American Journal of Mathematics, Vol. 60, No. 4) 60 (4): 897–936.

[47] A. Smola and B. Schölkopf. A Tutorial on Support Vector Regression. NeuroCOLT Technical Report NC-TR-98-030, Royal Holloway College, University of London, UK, 1998

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