INL/EXT-16-39860
Revision 0

# Time-Dependent Data Mining in RAVEN

Joshua Cogliati, Jun Chen, Japan Patel, Diego Mandelli, Daniel Maljovec, Andrea Alfonsi, Paul Talbot, Congjian Wang, Cristian Rabiti

September 2016


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

**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, does 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-16-39860
Revision 0

# Time-Dependent Data Mining in RAVEN

**Joshua Cogliati, Jun Chen, Japan Patel, Diego Mandelli, Daniel Maljovec, Andrea Alfonsi, Paul Talbot, Congjian Wang, Cristian Rabiti**

**September 2016**

**Idaho National Laboratory**
**Idaho Falls, Idaho 83415**

**http://www.inl.gov**

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

# Time-Dependent Data Mining in RAVEN

**INL/EXT-16-39860**
**Revision 0**

**September 2016**

**Approved by:**

| | |
|---|---|
| Name<br>Title [optional] | Date |
| Name<br>Title [optional] | Date |
| Name<br>Title [optional] | Date |
| Name<br>Title [optional] | Date |

# ABSTRACT

Risk Analysis Virtual Environment (RAVEN) is a generic software framework to perform parametric and probabilistic analysis based on the response of complex system codes. The goal of this type of analysis is to understand the response of such systems, in particular with respect to their probabilistic behavior, and to understand their predictability and drivers or lack thereof. The ultimate focus of such type of analysis is risk analysis.

Data mining capabilities are cornerstones to perform such deep learning of system responses. For this reason, static data mining capabilities were added to RAVEN in fiscal year 2015.

In real applications, when dealing with complex multi-scale, multi-physics systems, it seems natural that, during transients, the relevance of the different scales, and physics, would evolve over time. For these reasons, the data mining capabilities have been extended to allow their application over time.

This report describes the new RAVEN capabilities implemented with several simple analytical tests to explain their application and highlight the proper implementation.

This report concludes with the application of those newly-implemented capabilities to the analysis of a simulation performed using the Bison Implicit Simulation of Nuclear Fuels (BISON) code.

# CONTENTS



# FIGURES

## TABLES

# ACRONYMS

| | |
|---|---|
| BISON | Bison Implicit Simulation of Nuclear Fuels |
| DP | Dirichelet process |
| DBSCAN | density-based spatial clustering of applications with noise |
| EM | expectation maximization |
| GMM | Gaussian mixture model |
| PCA | principal component analysis |
| RAVEN | Risk Analysis Virtual Environment |
| TDM | Time-dependent data mining |

# Time-Dependent Data Mining in RAVEN

## 1. INTRODUCTION

The overall goal of the Risk Analysis Virtual Environment (RAVEN) software is to understand the probabilistic behavior/response of complex systems. In particular, RAVEN is focused on risk analysis. The capability to predict the likelihood that the system under consideration will end in a particular region of the output space, which is the base of the risk analysis, is already an available feature of the RAVEN code. Those types of analyses rely on the availability of probability distributions and sampling strategies, according to which the input space of the probabilistic systems are explored, and statistical post-processors used to understand the sources of dispersion of the system response.

In Fiscal Year 2015, RAVEN capabilities were extended to provide the tools to engineers to better understand the reasons/drivers of the system responses by adding advanced static data mining capabilities (e.g., clustering, principal component analysis [PCA], manifold learning) [1].

While the added capabilities showed a potential. it was clear that the static analysis was not sufficient to capture the complexity of system responses during transients. In fact, systems of interest, during transients, showed alternation of the physical phenomena characterizing the response. This made a static approach too narrow to characterize the system response. For this reason, during fiscal year 2016, the RAVEN infrastructure was extended to support data mining over time, or time-like variables.

Section 2 of this report addresses the basic and advanced static data mining capabilities that have been impacted by the new RAVEN infrastructure changes. Section 3 addresses the approach used to port such capabilities over the time domain. Finally, Section 4 discusses the simple analytical cases used to demonstrate the proper implementation of the algorithms and their use.

It is worth mentioning that given the high modularity of the RAVEN code, the creation of an infrastructure to analyze data over time adds this feature automatically to the already-existing post-processing capability beginning with the basic statistical relational analysis and ending with clustering and dimensionality reduction.

The report concludes with the application of the newly developed capabilities to a simulation performed using the Bison Implicit Simulation of Nuclear Fuels (BISON) code.

Appendix A includes the modification to the user manual as a consequence of this activity and Appendix B lists the new regression tests implemented.

## 2. EXISTING DATA MINING CAPABILITIES

Data mining is the computational process of discovering patterns in large datasets that involves methods from artificial intelligence, machine learning, statistics, and database systems. The overall goal of data mining is to extract information from a dataset and transform it into understandable structure.

This section reports briefly the mathematical formulation of the most relevant post-processing capabilities in RAVEN that the changes in the RAVEN code infrastructure extended from static to time dependent. The list ranges from the basic statistical relational analysis to the advanced techniques such as clustering and manifold learning. The full list is present in the updated RAVEN User Manual [2]. The examples in the following sections will use some additional methodologies, which will be introduced briefly directly in the example. One of the main reasons to report the mathematical formulation of the data post-processing capabilities that have been extended to the time dependent analysis is to provide guidance in the interpretation of the results shown at the end of this report.

## 2.1 Basic Statistical Relational Analysis

The items reported in this section are the basic statistical relational analyses that are used to characterize the relation embedded within the data set. Its use helps and augments the clustering and dimensionality reduction analysis.

This section presents the statistical methodologies that allow quantifying how strongly two (in the following generically referred to as *X* and *Y)* aleatory variable are related. The analysis presented is independent from *X* or *Y* being part of the stochastic system input space or output space. RAVEN does not make any specific distinction (except in a few cases where a special treatment might be needed) with respect to the belonging of *X* and *Y* to the input or output space. More specifically, depending from the belonging of *X* and *Y* to the input/output space, the analysis focuses on different relational aspects:

- Both *X* and *Y* belong to the input space—The relational analysis reveals correlation in the input space deriving, for example, from using multivariate distributions or biased sampling strategies
- One belongs to the input space and one to the output space—The relational analysis reveals correlations built by the system response function (the physical model of the system)
- Both belong to the output space—The relational analysis reveals correlations built by the system model among different responses monitored.

The covariance/variance analysis are a basic data mining capability, which were naturally extended to have time-dependent capabilities has consequence of the changes in the RAVEN infrastructure. Once paired with the clustering and dimensionality reduction capabilities, they are a useful tool to investigate the system behaviors.

***2.1.1.1 Covariance.*** In a simplistic sense, covariance measures how two variables vary together. In other words, it is a measure of correlation of relevant variables. The covariance is said to be positive/negative if the relevant variables vary congruently/inversely with respect to each other.

For two sets of random variables *X* and *Y* of *N* samples, the simplest formulation of the covariance matrix is:

$$\mathrm{cov}(X,Y)=\frac{1}{N}\sum_{i}(x_i-E(X))(y_i-E(Y)). \quad (1)$$

RAVEN possesses also the unbiased version and the weighted biased and unbiased versions.

***2.1.1.2 Correlation.*** Correlation informs how pairs of variables are related. The Pearson product-moment correlation coefficient matrix can be calculated from the covariance matrix as:

$$\Gamma(X,Y)=\frac{cov(X,Y)}{\sigma_x\sigma_y}. \quad (2)$$

2.1.1.3 ***Sensitivity.*** Sensitivity analysis is aimed to assess how sensitive two variables are to each other variation. They relate the ratio of rate of change of one function to the rate of change of another. It may be seen as somehow related to the derivative of one function with respect to another or as a slope. RAVEN uses sets of data and statistically measures sensitivities. The sensitivity option uses the least square linear regression for which that formula is:

$$\left(\hat{\beta}=(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}\right) \quad (3)$$

*where* $\hat{\beta}$ is the unknown parameter in the regression. Note that RAVEN repeats the process for each target variable so that all the variables are used as both **X** and **Y**.

2.1.1.4 ***Variance-Dependent Sensitivity.*** Variance-dependent sensitivity matrix is the matrix of sensitivity coefficients that show the relationship of the individual uncertainty component to the standard deviation of reported value for a test item. The variance-dependent sensitivity matrix VS is calculated for datasets X and Y as:

$$L(Y|X) = \mathbf{E}(Y) + cov(X,Y)vc^{-1}(X)[X - \mathbf{E}(X)]. \quad (4)$$

From which the sensitivity coefficient is given by:

$$cov(X,Y)vc^{-1}(X) \quad (5)$$

where $vc(X)$ is the symmetric matrix with $(var(X_1), var(X_2), \dots, var(X_n))$ as the diagonal. Note that RAVEN repeats the process for each target variable so that all the variables are used as both *X* and *Y*.

## 2.2 Clustering

A loose definition of clustering is the grouping of objects with similar properties. Closeness according to some predefined metric is the main property we look at to define clusters. Several closeness metrics exist in literature [3,4,5]. These different metrics lead to different clustering algorithms. In this section, we will discuss several such clustering algorithms currently available in RAVEN.

The list of the clustering algorithms that has been ported to be time dependent includes the following:

- Affinity Propagation
- Density- Based Spatial Clustering of Applications with Noise (DBSCAN)
- Gaussian Mixture Model with Dirichlet Process
- Gaussian Mixture with Expectation Minimixation
- Gaussian Mixture with Variable Bayesian
- K-Means
- Mean Shift
- Mini Batch K-Means
- Spectral Clustering (applies K-means in a reduced dimensionality space that is based on an embedding of the affinity matrix of the dataset).

Except for the spectral clustering. which theory is very similar to the K-means, a short theoretical explanation of the different clustering algorithms is provided below.

### 2.2.1 K-Means

Given a dataset, *D* of size *N*, the K-means algorithm separates *D* into *K* clusters, $C_k$ where *k*=1,2,3...*K* such that all *K* clusters have equal variance. Here, *K* must be specified beforehand. This K-means algorithm is also known Lloyd's algorithm. The main aim of this algorithm is to choose centroids such that the inertia or within-cluster sum of squares criteria:

$$\sum_{i=0}^{K} \left( \|x_k - \mu_i\|^2 \right), \quad (6)$$

is minimized,

*where*

$\mu_i$ = mean value of samples (centroid) within respective clusters

$x_k$ = sample value in a particular cluster $C_k$.

The algorithm is as follows:

1. Choose the initial *K* centroids (within-cluster mean value).
2. Assign the sample to the nearest centroid. Often, Euclidean distance is used to determine the nearest centroid.
3. Upon assignment of each sample to its respective centroid, new centroids are calculated for each cluster. The means of the clusters are often chosen to be the new centroids.
4. Upon assignment of new centroids, check for convergence. In other words, check and see if the centroids are still moving by checking if the difference between the new and the old centroids is below a predefined tolerance.

This algorithm essentially has two drawbacks:

1. The minimization criteria makes an assumption that clusters are convex and isotropic. This is not always true.
2. The minimization criteria is not normalized. Thus with increasing dimension, where the Euclidean distances become inflated, this is not a criterion.

In RAVEN, most of the data-mining methods are available through Scikit-learn [6]. The K-means method can be accessed in RAVEN through the "KMeans" post-processor.

Another way to access the K-means algorithm is through "MiniBatchKMeans." This uses subsets of original input data, randomly sampled in each training iteration. The aim is to reduce the computational time taken by the K-means algorithm while still optimizing the same function. Just like the original K-means algorithm, the mini-batch K-means algorithm is as follows:

1. Form mini-batches by randomly choosing *n* samples.
2. Assign these samples to the nearest centroid.
3. Update centroids by taking streaming average of sample and all previous samples assigned to the centroid.
4. Check for convergence by seeing if the centroids are still moving. Iterate until convergence.

This method decreases the rate of change of centroid over time, which helps accelerate convergence (for that given convergence criteria). However, this is not a true acceleration scheme since the results or the original method are not necessarily reproduced. It has been observed that this method produces different, less accurate results than the standard K-means algorithm.

### 2.2.2 Affinity Propagation

Affinity propagation is an exemplar-based clustering algorithm. Exemplars are a set of samples that represent other samples the best based on a given criterion. This method simultaneously considers all samples as exemplars. It sends messages between pairs of samples that eventually determine the clustering pattern. Two kinds of messages are primarily sent to determine the suitability of a given sample to be the exemplar of the other:

1. Responsibility $R(i,k)$—Accumulated evidence that a sample $k$ should be the exemplar for sample $i$. Responsibility function, $R(i,k)$,is:

$$R(i,k) \leftarrow S(i,k) - \max\left(A\left(i,\tilde{k}\right) + S\left(i,\tilde{k},\forall\tilde{k} \neq k\right)\right) \quad (7)$$

*where*

$S(i,k)$ = pairwise similarity function that defines how well sample $k$ represents sample $i$.

2. Availability $A(i,k)$ —Accumulated evidence that sample $i$ should choose sample $k$ to be its exemplar. Availability function, $A(i,k)$, is:

$$A(i,k) \leftarrow \min\left(0, R(k,k) + \sum_{(\tilde{\imath}|(\tilde{\imath} \notin i,k))} R(\tilde{\imath},k)\right) \quad (8)$$

Therefore, exemplars must be similar to several samples and must be chosen by several samples to represent them. Roughly, the affinity propagation algorithm can be summarized as follows [7]:

1. Define a set of pairwise similarity functions, $S(i,k)$. This function can be user defined. In general, squared Euclidean distance is chosen to be the similarity function. Choose all samples as exemplars to initially have responsibility equal to zero.
2. Set all availabilities to zero.
3. Calculate the response and availability function for all samples.
4. Calculate $\widetilde{C_i} = \max_k(A(i,k) + R(i,k))$ where $i$=1,2,...$N$. This is where an examplar is assigned to sample i. In other words, if for sample $i$, sample $k$ has the maximum availability-plus-responsibility value, then sample $k$ will be the examplar of sample $i$.

In case there are contradictory results, one K-medoids (K-means but with medoids instead of means) run might be necessary. Dueck 2009 [7] presents an in-depth presentation of this algorithm. In RAVEN, this algorithm is accessible through the "AffinityPropagation" post-processor.

### 2.2.3 Density-Based Spatial Clustering of Applications with Noise

Clusters are formed by identifying high sample density areas separated from low sample density areas. The DBSCAN clusters can be of any arbitrary shape. The main concept behind this algorithm is that of core samples. Core samples are the areas of high sample density. These are defined using two parameters—cardinality, $d$ and distance ε. Higher cardinality and lower distance indicate higher sample density required to form clusters.

Formally, a core sample is a sample in the dataset with cardinality greater than or equal to $d$ with distance ε. The samples inside distance ε are called neighbors. A cluster is then built from core samples recursively by taking core samples, finding all neighbors that are core samples, finding their neighbors that are core samples, and so on. The cluster also incorporates samples that are not core samples themselves but are neighbors of core samples.

The DBSCAN algorithm may be summarized as follows:

1. Input d and ε.
2. For a given dataset, determine core samples $C_i$ and neighbors.
3. Take union of core samples $C_i$ and $C_j$ if $C_i \cap C_j \neq \varphi$ over all core samples.

Step 3 of the above algorithm will return separated clusters. Note that this will not always be able to cluster all the points since some points can be too far from other points, and so this algorithm can return outliers. In RAVEN, this algorithm can be accessed using the "DBSCAN" post-processor.

### 2.2.4 Mean-Shift

This clustering algorithm is a nonparametric algorithm that does not require the number of clusters as an input. It automatically selects the number of clusters based on the bandwidth—the size of regions to search through and number of local minima that result from the bandwidth specification. It is a centroid-based algorithm that works by updating centroids to be the mean of points within a given region.

Assuming the bandwidth returns a neighborhood $N(x_i)$ around centroid $x_i$, the mean shift vector, $s(x_i)$, is defined as follows:

$$s(x_i) = \frac{\sum_{x_j \in N(x_i)} K(x_j - x_i) x_j}{\sum_{x_j \in N(x_i)} K(x_j - x_i)}, \tag{9}$$

*where*

$K(x_j - x_i)$ = kernel function used for the weighted mean.

This mean shift vector is computed for each centroid that points towards the region of maximum increase in the density of points. Now given a centroid $x_i$ for iteration, $k$, the updates are given as:

$$x_{i,k+1} = x_{i,k} + s(x_{i,k}). \tag{10}$$

The mean shift algorithm can be summarized as follows:

1. Initialize with a random centroid.
2. Assign members to this centroid based on bandwidth.
3. For points unassigned, choose other centroids at random until each point is assigned to a centroid.
4. Calculate the mean of each neighborhood's $x_i$.
5. Shift the mean by mean shift vector $s(x_i)$.
6. Calculate the members in the new neighborhoods.
7. Check for convergence. When centroids move less than specified tolerance, the method is converged.

Detailed explanation of theory and applications can be found in [8] and [9]. In RAVEN, this functionality is provided using the "MeanShift" post-processor.

### 2.2.5 Gaussian Mixture Model Based Clustering

***2.2.5.1 Gaussian Mixture Model.*** For a given dataset, D, Gaussian mixture model (GMM) assumes that all the samples in D can be generated from the linear combination of a finite number of Gaussian distributions of unknown parameters. According to the Scikit-learn manual, this model may be thought of as a generalization of K-means clustering to incorporate information about the covariance of D as well as centers of hidden (or latent) Gaussians.

These Gaussians of unknown parameters can be estimated using different methods: expectation maximization (EM), variational Bayesian, and Dirichelet process (DP). A detailed explanation of the GMM algorithm with EM can be found in Russell and Norvig [10]. A brief summary of their presentation is provided below.

Let the mixture distribution that presumably generates the dataset be *D*. Distribution *D* has *n* Gaussian components. A data point may be generated by choosing a component Gaussian and sampling

from that component. If $C_i$ is a component of $D$, where $I = 1, 2,...N$, then the mixture distribution is given by the following linear combination:

$$D(x)= \sum_{i=1}^{N} w_i C_i. \tag{11}$$

*where*

$i_{th}$ = Gaussian component, $C_i$, of mixture Distribution $D$ has parameters $\mu_i$ and covariance $\Sigma_i$.

Now, given a dataset, the clustering problem is to recover a GMM from a given dataset. It is not known beforehand what component of the GMM produces which sample from the dataset. The parameters for constituent components of the GMM are also unknown. Thus, there must be a method to estimate these. This is where the different estimation methods come in. First, we look at EM.

***2.2.5.2 Expectation Maximization.*** First, we assume that we know the components of the GMM (generated at random—they may be centered on a sample or learned using K-means), then for each sample in the dataset, we find the probability that the sample belongs to each component. After that, each component is fitted to the entire dataset and each sample is weighted by the probability that it belongs to that component. We iterate on this process until convergence (i.e., parameters of components no longer change). This algorithm may be summarized as follows [10]:

1. Initialize the GMM parameters.
2. Compute probability, $p_{i,j}$, that sample $x_j$ was generated using component $C_i$.
3. Compute the effective number of data points currently assigned to $C_i$, $n_i = \sum_j p_{i,j}$.
4. Compute new mean, covariance, and component weights:

$$\mu_i \leftarrow \sum_j \frac{p_{i,j} x_j}{n_i} \tag{12}$$

$$\Sigma_i \leftarrow \sum_j \frac{p_{i,j}(x_j-\mu_i)(x_j-\mu_i)^T}{n_i} \tag{13}$$

$$w_i \leftarrow \frac{n_i}{N} \tag{14}$$

5. Check for convergence of parameters and repeat Steps 2 through 4 above until convergence.

Each component returns a cluster. The clustering analysis based on GMM and expectation Maximization can be accessed in RAVEN through the "GMM" post-processor.

***2.2.5.3 Dirichelet Process.*** The DP does not require the number of clusters as an input. It is a prior probability distribution on clustering with an infinite number of components. A classic way of explaining how this works is by using the analogy of a Chinese restaurant process. Think of a Chinese restaurant with an infinite number of empty tables initially. The first customer takes the first table. After that, as the customers arrive, they either join the preoccupied table with a probability proportional to the occupancy of that table or they sit on a new table with a probability proportional to the concentration parameter α. After a known number of customers sit, we see that only a finite number of tables have been

occupied. The higher the α value, the higher the number of tables occupied. Here, for the DP GMM, each component of the mixture is analogous to the table and each sample is analogous to the customer.

Variational inference techniques for the DP work with finite approximations. Inference using Gibbs sampling is a common way of inferring mixture components. For exact algorithm and mathematical derivation, refer to [11].

The clustering analysis based on GMM and the DP process can be accessed in RAVEN through the "DPGMM" post-processor.

This functionality is available in RAVEN through the "DPGMM" post-processor.

***2.2.5.4 Variational Bayesian.*** The GMM may be implemented with variational inference algorithms. Variational inference GMM chooses Gaussians with random initial parameters and minimizes the lower bound of model evidence instead of data likelihood. Model evidence incorporates prior guesses. Just like with expectation Maximization, the algorithm alternates between finding probability for each point to be generated by each mixture component and fitting the mixtures to these assigned points. Variational methods add regularization by integrating information from prior distributions. Roughly, this algorithm can be summarized as follows [12]:

1. Given a dataset, initialize parameters and define the number of clusters.
2. Define variational distributions: variational multinomials, variational Gaussians.
3. Update the variational multinomial.
4. Update the mean and variance of variational Gaussian.
5. Check for convergence of evidence lower bound.

For a detailed mathematical derivation of the algorithm, refer to [12] and [13].

The clustering analysis based on the GMM and variable Bayesian can be accessed in RAVEN through the "VBGMM" post-processor.

## 2.3 Dimensionality Reduction

Dimensionality reduction allows the number of dimensions in the problem to be reduced, by transforming to a new coordinate system. RAVEN includes a variety of methods for doing this including PCA, truncated singular value decomposition (differs from PCA only in the algorithm and the number of retained vectors) independent component analysis and practically all the unsupervised learning algorithms available in Scikit-learn [6]. These algorithms have to goal to remove redundancy in the data set to make its interpretation easier.

The list of the dimensionality reduction algorithms made available from Scikit-learn and embedded in the new time-dependent development of the RAVEN infrastructure is as follows:

- Exact Principal Component Analysis
- Fast Independent Component Analysis
- Kernel Principal Component Analysis
- Locally Linear Embedding
- Mini Batch Sparse Principal Component Analysis
- Multi- Dimensional Scaling
- Randomized Principal Component Analysis
- Sparse Principal Component Analysis

- Truncated Singular Value Decomposition.

The theory of all the dimensionality reduction algorithms could be found in Scikit-learn [6] while the approach to extend their applicability at the time-dependent domain is explained in Section 3. In the following, an exemplification of the theory of the PCA will be presented since it is used in the examples in Sections 4 and 5.

### 2.3.1 Principle Component Analysis

PCA is used to analyze correlation between sets of variables. To fix ideas we could consider a set of point in a two-dimensional plane as shown in Figure 1. We can notice that the data are mainly dispersed in the direction of the longest arrow while the dispersion in the direction of the smallest arrow is minimal. PCA has exactly this task, it determines a set of orthogonal directions where the first one is the one along which the variance of the data is the highest and the last one is the one with the lowest marginal variance [14]. The first direction (component) could be found by:

$$w_1 = \arg\max_{\|w\|=1}\{\sum_i (x_i \cdot \mathbf{w})^2\}. \tag{15}$$

*where*

$w$ = weight or loading

$w_1$ = first component

$x_i$= zero centered data point (original value minus the mean).

The first component's effect is subtracted out, and then the next component is found. It could be proved that those directions are the eigenvectors of the covariance matrix of the data and the ordering (from the one which explains the most of the variance to the least) is given by the magnitude of the eigenvalues. This provides the most common way to extract the PCA from a dataset.

The information that could be extracted for the PCA analysis is as follows:

- The largest component of the eigenvector are strongly related variables (also known as redundant). This is usually referred as the load of the component
- The size of the corresponding eigenvalue or equivalently the amount of explained variance is a measure of how much a direction explains the dispersion of the dataset
- The measure of the cumulative explained variance (sum of the explained variance by each component starting from the one corresponding to the largest eigenvalue in decreasing order) could be used to determine how many dimensions are necessary to represent the dataset, up to a certain approximation on the explained variance.

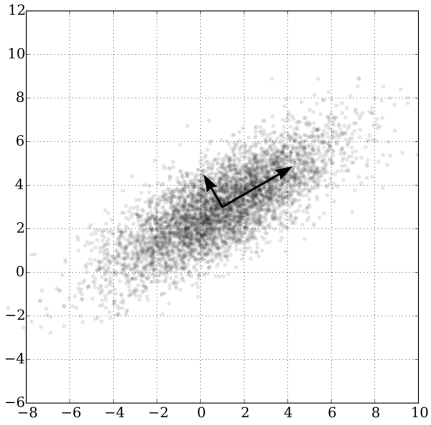


Figure 1. Principle component analysis exemplification (source: https://commons.wikimedia.org/wiki/File%3AGaussianScatterPCA.svg, By Nicoguaro (Own work) [CC BY 4.0 (http://creativecommons.org/licenses/by/4.0)], via Wikimedia Commons).

# 3. EXTENSION TO THE TIME-DEPENDENT DATA MINING

## 3.1 Time Slicing

For the time-dependent data mining (TDM), the data needs to be related to a 'time-like' variable. This can be any monotonically increasing variable in the input or output of the simulation. RAVEN can interpolate so that all the data are synchronized to be on a common time index where for each time step all the data are available. For example, if the data from Simulation 1 is available at Times A, C, and E and data from Simulation 2 are available at Times B, C, and D, then the data from Simulations 1 and 2 need to be interpolated to Times A, B, C, D, and E or some other common set of time steps. RAVEN allows the time steps to be chosen in two different ways. The first is the intersection of all points. The second is that the user specifies the number of points desired and a grid is created and the data are interpolated to the grid. The data are linearly interpolated. The statistics or data mining operation is then run over the data for each of the common set of time steps.

## 3.2 Inertia-Driven Dampening

After implementing the time slicing infrastructure in RAVEN, practically all the RAVEN post-processors are automatically migrated to become time dependent. While this is very convenient and allows a steep extension of RAVEN capabilities, it faces some challenges when applied directly to clustering and dimensionality reduction algorithms.

Focusing on the clustering algorithms, they could be used at each time step but the lack of temporal connection between the different time steps might induce unnatural fluctuation of the cluster labeling.

To fix the idea, we could consider a clustering algorithm identifying two clusters. The algorithm applied at two consecutive time steps, $t_1$ and $t_2$,identify two centroids (labeled A and B). Since there is no time memory, there is no assurance that Centroid B at $t_2$ is the one that relates to (the most likely evolution of) Centroid B at $t_1$.

See for example, in Figure 2, where for each time step, the K-means algorithm with two clusters is used, and the desired cluster labels are marked in red in the figure while the one provided by the code is marked in blue. In fact, in this example, the clustering algorithm will correctly divide the data into two clusters at each time step, as desired. However, the labeling for each cluster may not stay the same along the evolution of time. One possible scenario is marked in the figure as red. In this case, the labeling of the

cluster flips at time instance 2.0. Another example is shown in Figure 3, where the cluster centroids are plotted. As can be seen, different clusters along the time are labeled the same.

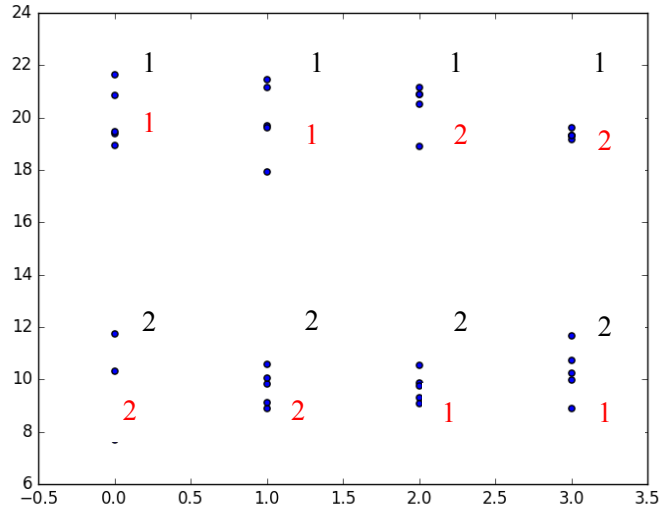


Figure 2. Two cluster example.

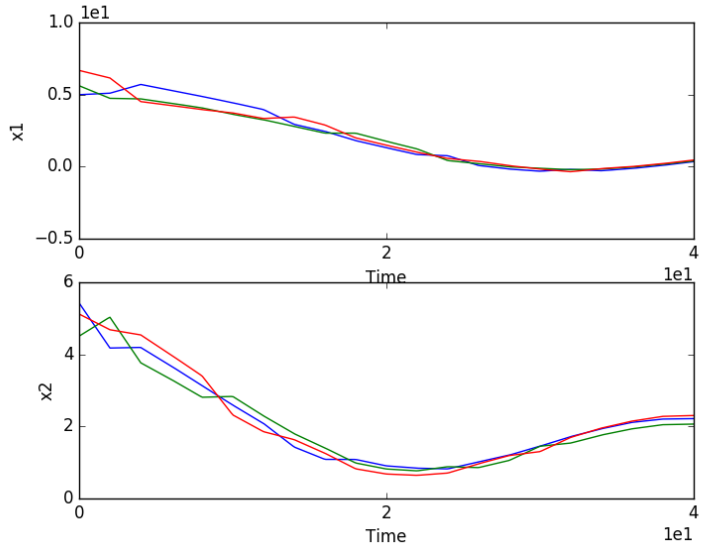


Figure 3. Cluster centers.

To address this issue, in the current implementation, it is assumed that clusters along the time are changing slowly, despite the evolvement brought by system dynamics. In other words, we consider the cluster has inertia that prevents disruptive changes and therefore it is possible to perform a cluster association crossing time step using distance metrics applied between the cluster centroids. Based on this assumption, several mapping mechanisms are implemented to post-process the labels returned by clustering algorithms. The idea is to define a certain distance metric between cluster i at time t and cluster j at time t-1, denoted as *dist(i,j)*, and the goal is then to find the set of exclusive pairs $M=\{(i,j)\}$ such that each cluster i for time step t appears exactly once and each cluster j for step t-1 appears exactly once, and the global distance is minimal (i.e., $\sum_{(i,j)} dist(i,j)$ is minimized). Finally, cluster i at time t and cluster j at time t-1 are labeled as the same cluster if and only (i,j) is in *M*. Currently, four distance metrics are implemented; they are briefly described in the following sections.

### 3.2.1 Euclidean Distance-Based Metric

In this case, the distance between cluster i at time t and cluster j at time t-1 is measured by the Euclidean distance between their centroids. Let $x_1$ be the centroid of cluster i at time t and $x_2$ be the centroid of cluster j at time t-1. The distance between these two is given by:

$$dist_{ij} = \sqrt{(x_1 - x_2) \cdot (x_1 - x_2)} \tag{16}$$

### 3.2.2 Membership-Based Metric

In this case, the distance between cluster i at time t and cluster j at time t-1 is measured by the joint members (i.e., the number of histories that is in cluster i at time t and cluster j at time t-1). Let Point1 be all the members of cluster i at time t and Point 2 be all the members of cluster j at time t-1. The distance is measured by:

$$dist = -\text{size}(point_1 \cap point_2) \tag{17}$$

Note here the minus sign is added so that the more joint members there are the smaller the distance is.

### 3.2.3 Euclidean Distance with Variance-Based Metric

In this case, the variance of each cluster is also assumed to have "inertia." The distance between cluster i at time t and cluster j at time t-1 is then measured by the Euclidean distance between their centroids plus the difference of the empirical variances. Let $x_1$ be the centroid of cluster i at time t and $x_2$ be the centroid of cluster j at time t-1, and $v_1$ be the empirical variance of cluster i at time t and $v_2$ be the empirical variance of cluster j at time t-1. The distance is measured by:

$$dist_{ij} = \sqrt{(x_1 - x_2) \cdot (x_1 - x_2)} + |v_1 - v_2| \tag{18}$$

### 3.2.4 Euclidean Distance with Decay

In this case, the distance between cluster i, at time t, and the label j is computed based on labels K steps back in time (or any other pivot variable). Let x21, x22, … , x2K be the centroids of cluster j at time t-1, t-2, … , t-K. The distance between cluster i at time t and the label j is measured by:

$$dist_{ij} = \sum_{k=1}^{K} \sqrt{(x_1 - x_2) \cdot (x_1 - x_2)} e^{-(k-1)R} \tag{19}$$

In other words, it is the sum of the Euclidean distances between cluster i at time t and cluster j at time t-k for k from 1 to K. Note that the contribution to distance by cluster j at time t-k is decayed by the exponential term with decaying rate defined as R (decRate in the code).

Note that currently this metric is the default to re-map the labels returned by Scikit-learn, with default values as K=5 and decRate = 1.0.

### 3.2.5 Results

Figure 4 plots the centroids along the time for Figure 3 after using the re-map function implemented (default options: Euclidean distance with decay, k=1 decay rate=1.0). As can be seen, the cluster labels are well recognized over different time steps.

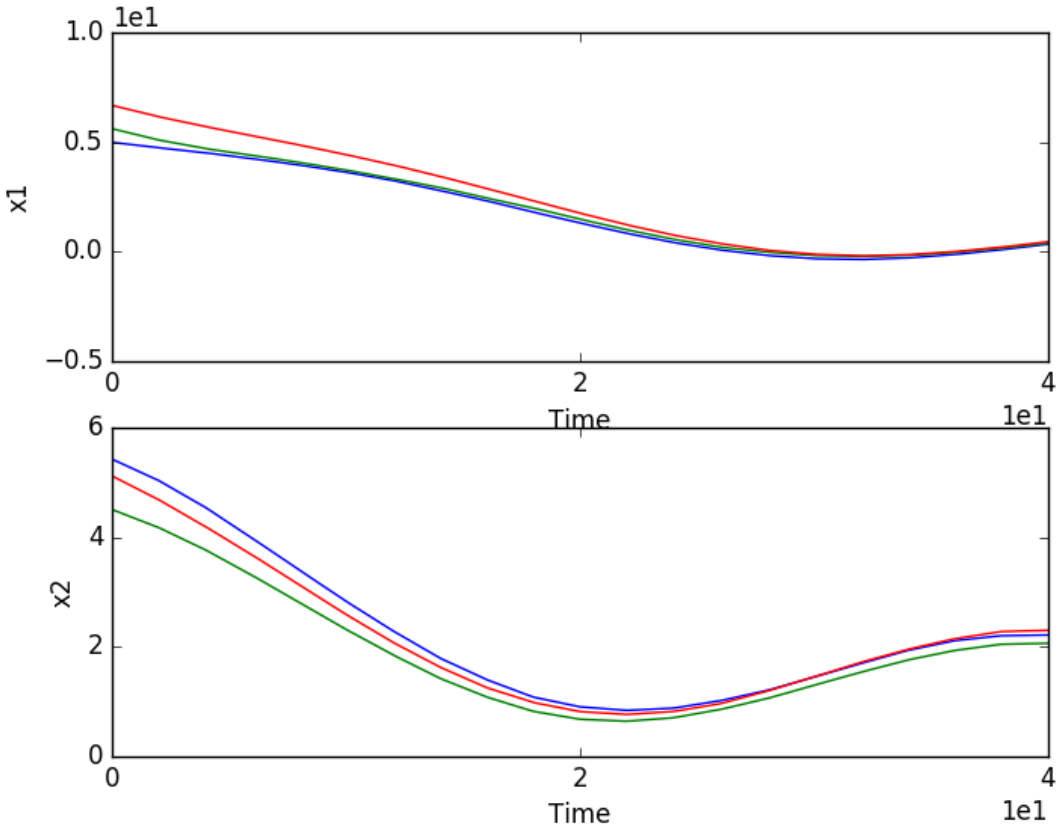


Figure 4. Remapped cluster centers.

# 4. THEORETICAL EXAMPLES

## 4.1 Cluster Example

This model was created for testing clustering alone. The model equations are as follows:

$$x(t) = x_0(u) + \tilde{x} + t(v_x(u) + \tilde{v}_x)$$

$$y(t) = y_0(u) + \tilde{y} + t(v_y(u) + \tilde{v}_y)$$

$$\tilde{x} \sim N(\mu = 0, \sigma = 0.1),\ \tilde{y} \sim N(\mu = 0, \sigma = 0.1)$$

$$\tilde{v}_x \sim N(\mu = 0, \sigma = 0.01),\ \tilde{v}_y \sim N(\mu = 0, \sigma = 0.01)$$

$$u \in \{1,2,3\}, P(u) = \left\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right\} \text{ (Discrete variable)}$$

$$\big(x_0(1), y_0(1)\big) = (2,2)$$

$$\big(x_0(2), y_0(2)\big) = (4,1)$$

$$\big(x_0(3), y_0(3)\big) = (4.5,4)$$

$$\Big(v_x(1), v_y(1)\Big) = (0.1,0.1)$$

$$\Big(v_x(2), v_y(2)\Big) = (0.05,0.2)$$

$$\Big(v_x(3), v_y(3)\Big) = (-0.05, -0.2)$$

The simple clustering model creates 100 points with the base location and initial velocity sampled from three different groups (the groups are determined by the sampling over the discrete variable u). The initial location and velocity of each point are created by using the base values for the group and then adding values sampled from normal distributions centered at 0.0 and with sigma 0.1 ($\tilde{x}, \tilde{y}$) for the location and 0.01 for the velocity ($\tilde{v}_x, \tilde{v}_y$). Then, as the time increases, each point heads in its direction determined mainly by the characteristic group velocity ($v_x(u), v_y(u)$). Two of the groups are headed towards each other. This model provides a clear and simple test case for use with clustering.

First, the DBSCAN method was used. For DBSCAN, the method automatically tries to find different clusters. As can be seen from the plots in Figure 5, initially at time=0.0, as expected, three distinct clusters were found on correspondence of the initial coordinate of each group. As two of the clusters approach each other, the clustering algorithm struggles and creates more clusters while it is trying to maintain a balanced degree of separation among the clusters. This can be seen in Figure 6 where the centroids of each cluster are graphed. Figure 5 shows also a labeling -1, which is assigned to outliers.

Secondly, the K-means method is tested. For K-means, the number of clusters is an input parameter and therefore, it is fixed to 3 since we provide initially three differentiated clusters. As expected, the right three clusters are found at time=0.0. Because K-means has the number of clusters to find as an input parameter, it continues to find the original clusters, throughout the model, with some mixing when the clusters collide as shown in Figure 7. The colliding of the clusters can be seen in the centroid graph in Figure 8.

This example is built with the purpose to illustrate the behavior of the clustering algorithms in a difficult situation where the time evolution brings clusters to collide creating an unclear situation with respect the separation of the clusters. This effect needs a high degree of attention from the user that can mitigate the effect controlling the cluster memory model and parameter.

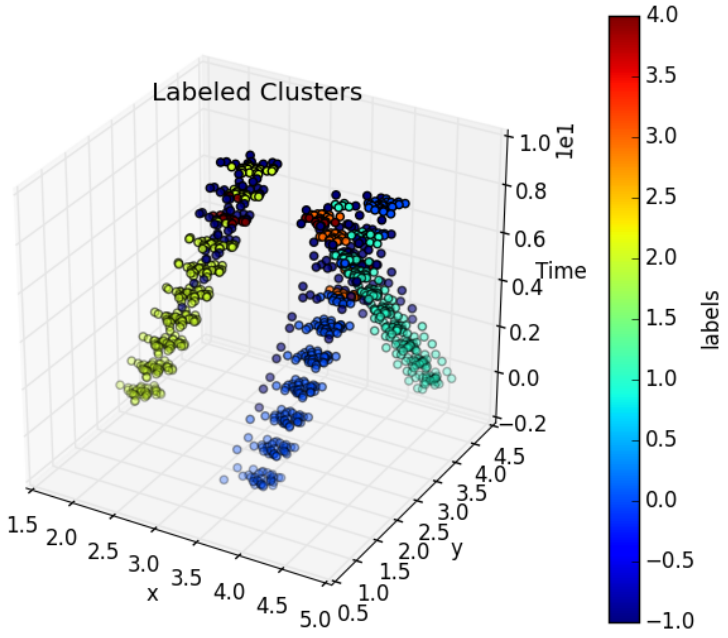


Figure 5. Density-based spatial clustering of applications with noise clusters.

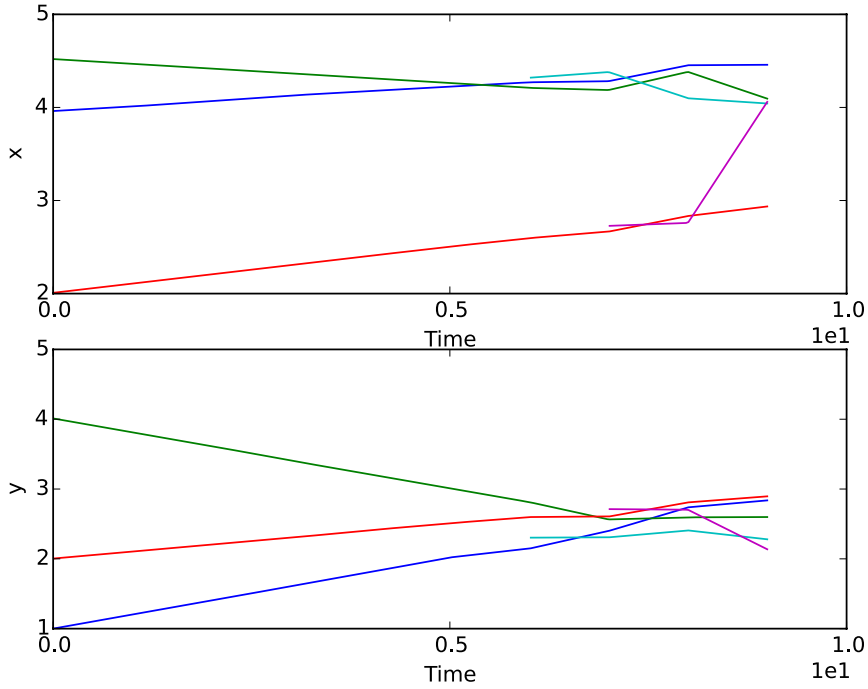


Figure 6. Density-based spatial clustering of applications with noise centroids.

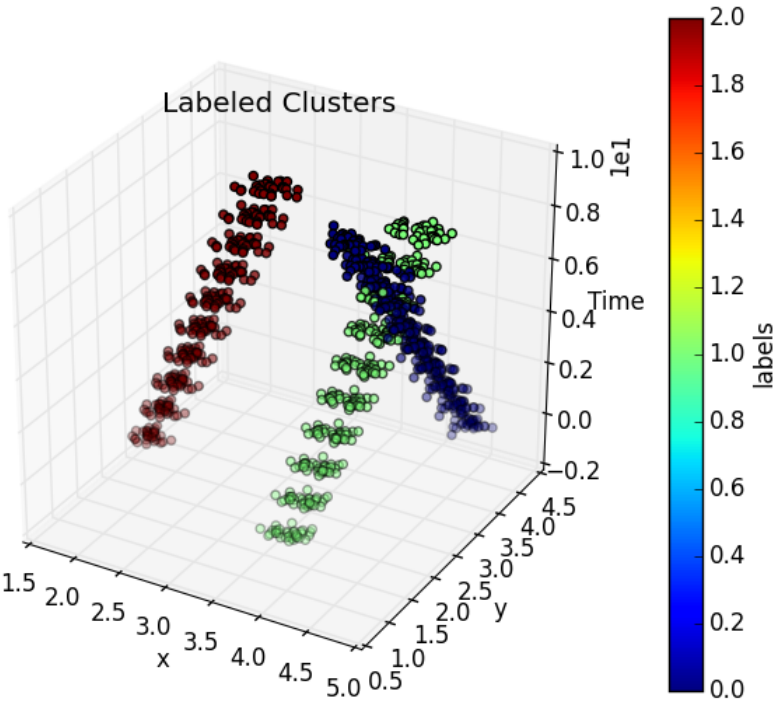


Figure 7. K-means clusters.

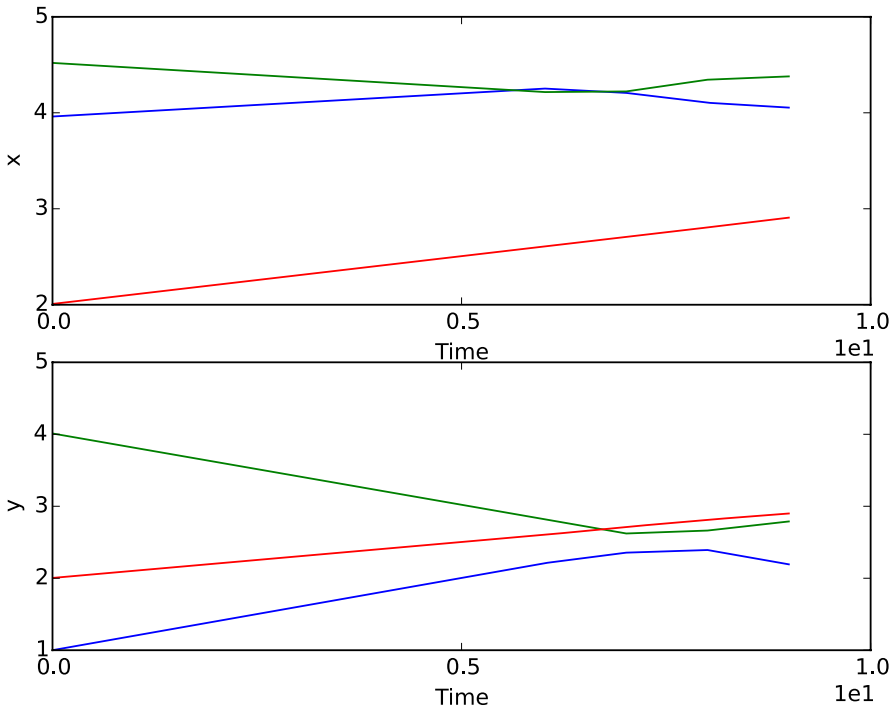


Figure 8. K-means centroids.

## 4.2 Midway Shift Example

This example shows the output of the simulation shifts halfway as a function of time. This example has been chosen to show a sharp change in the relation analysis as it could happen in a physical model if the dominant physical phenomena would change during a transient analysis. The equation representing the model is implemented in RAVEN as "external model" (directly interfaced with RAVEN) where the basic form is:

$$z_1[t] = a(t)x$$

$$z_2[t] = b(t,x)y$$

$$z_3[t] = c(t,x)(x+y)$$

The values of the parameters $a(t), b(t,x), c(t,x)$ are:

$$a = \left(0.5 - \frac{\arctan(t-20)}{\pi}\right) + 5\left(0.5 + \frac{\arctan(t-20)}{\pi}\right)$$

$$b = (H(x)+3)\left(0.5 - \frac{\arctan(t-20)}{\pi}\right) + (2H(x)+4)\left(0.5 + \frac{\arctan(t-20)}{\pi}\right)$$

$$c = (1.5H(x)+0.5)\left(0.5 - \frac{\arctan(t-s)}{\pi}\right) + (-1.5H(x)-0.5)\left(0.5 + \frac{\arctan(t-s)}{\pi}\right)$$

In addition to the halfway shift, this example presents a clear separation in two clusters with respect to x (at least at the beginning) due to the usage of the Heaviside function $H(x)$. The variable *x* is normally distributed centered at 0.0 with sigma 1.0 and the *y* variable is normally distributed centered at 0.0 with sigma 2.0.

A set of representative cases (*x* is either -1 or 1, and *y* is either -1 or 1) is shown in Figure 9. For $z_1$, since it is only dependent on *x,* and *t*, there are two lines, one for *x* = -1 and one for *x* = 1.0. For $z_2$, *x* chooses the *b* value, and *y* can be -1 or 1, so there are four combinations so four lines. For $z_3$, there are only three lines since the couples $(x = 1, y = -1)$ and $(x = -1, y = 1)$ lead to the same value of $(x + y)$.

The covariance were calculated with respect to time and are reported in Figure 10 to show the impact of the shift at t=20 in the overall system response.

K-means with two groups was used to cluster the data. Figure 11 shows the centroids found, and Figure 12 shows the clusters found in the $z_1$ and $z_2$ space.

To be more precise, the clustering was performed with respect to the variables $z_1$, $z_2$, and $z_3$. Figure 11 shows the time evolution of the centroids projected separately in each dimension ($z_1$, $z_2$, and $z_3$). The use of the Heaviside function generated a clear cluster separation that was easily captured by the K-means.

In order to highlight how the system has a sharp shift in its behavior at t=20.0, DBSCAN was also tested. DBSCAN has the capability to independently decide the number of clusters needed. Figure 13 shows the centroids found, clearly illustrating that DBSCAN identifies the change in the behavior of the systems and introduces a second cluster. Figure 14 shows the time histories time evolution colored with respect to their cluster label and projected in the $z_1$ and $z_2$ space ($z_3$ has been suppressed). Note that DBSCAN does not always classify all the points and outliers are labeled with -1.

This example first illustrated how the relational analysis is helpful to identify changes in the system behavior, more specifically how it could be used to identify regions of a transient where the relevance of physical phenomena changes. This is in fact what the relational analysis reveals. We could in fact think that the system could be approximated by two different linear representation before and after t=20.0 corresponding to two different dominant physical phenomena. The relational analysis reveals when this transition take places and provides suggestion concerning which new phenomena is becoming relevant by highlighting a shift in the relevance of the input parameters toward the determination of the values of the system figure of merits.

The clustering exercise also provided insight on the behavior of the system. Especially the DBSCAN, which was able to detect a bifurcation in the system behavior adding a specific cluster.

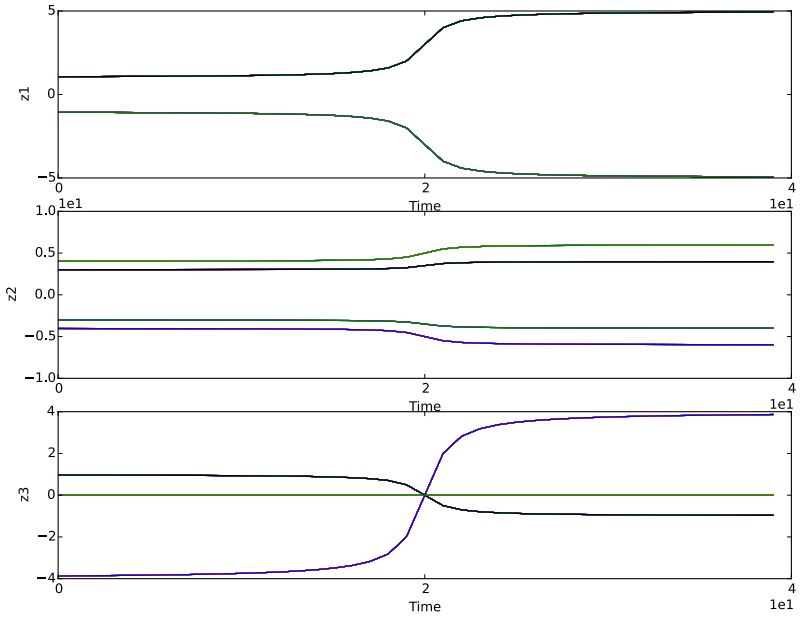


Figure 9. Representative midway shift examples.

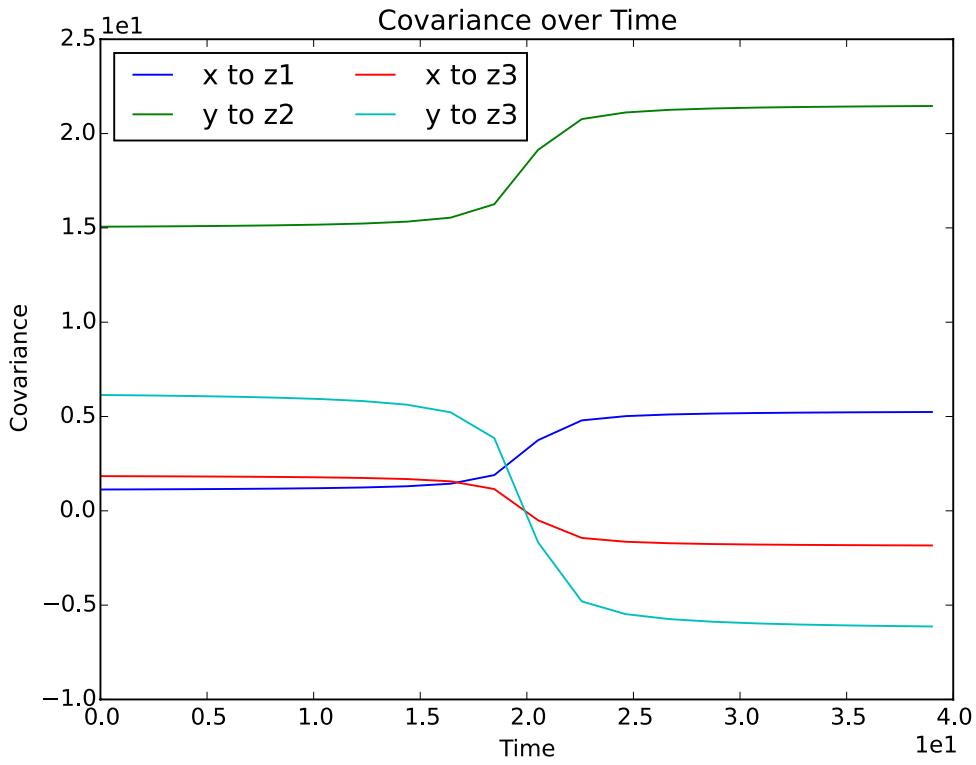


Figure 10. Covariance over time.

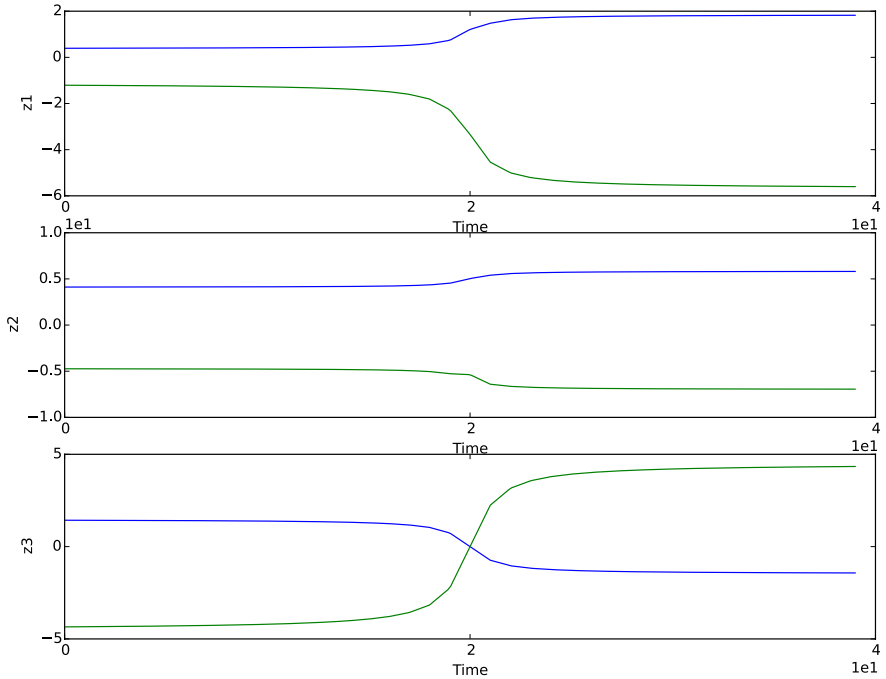


Figure 11. Centroids found with K-means.

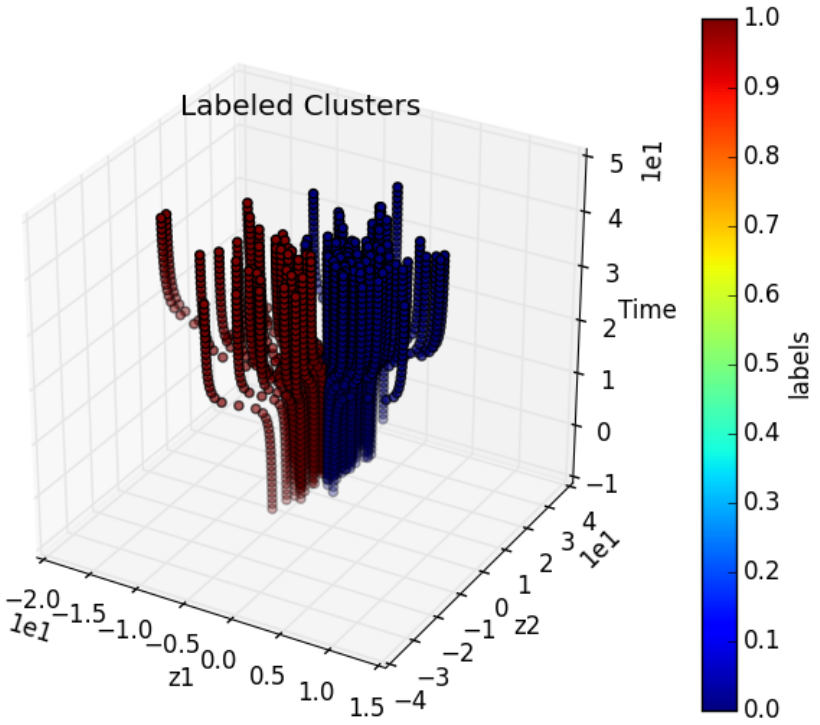


Figure 12. Clusters found with K-means.

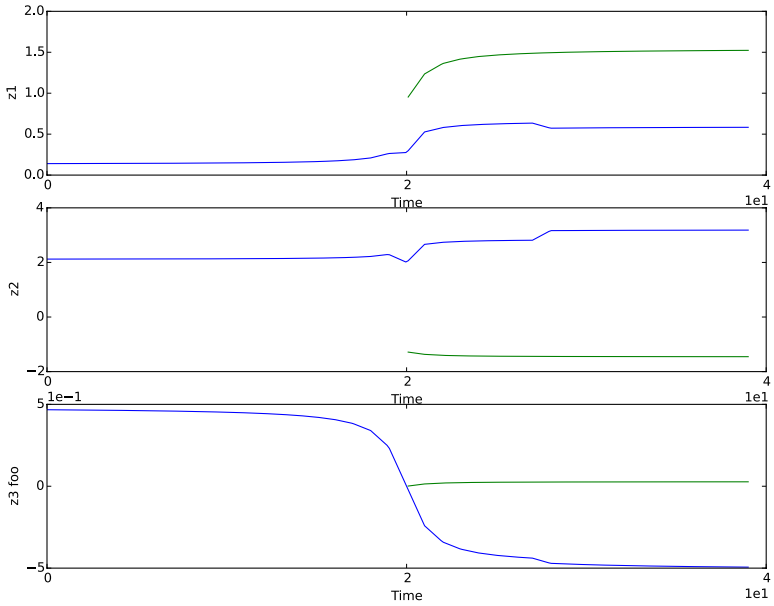


Figure 13. Centroids found with density-based spatial clustering of applications with noise (DBSCAN).

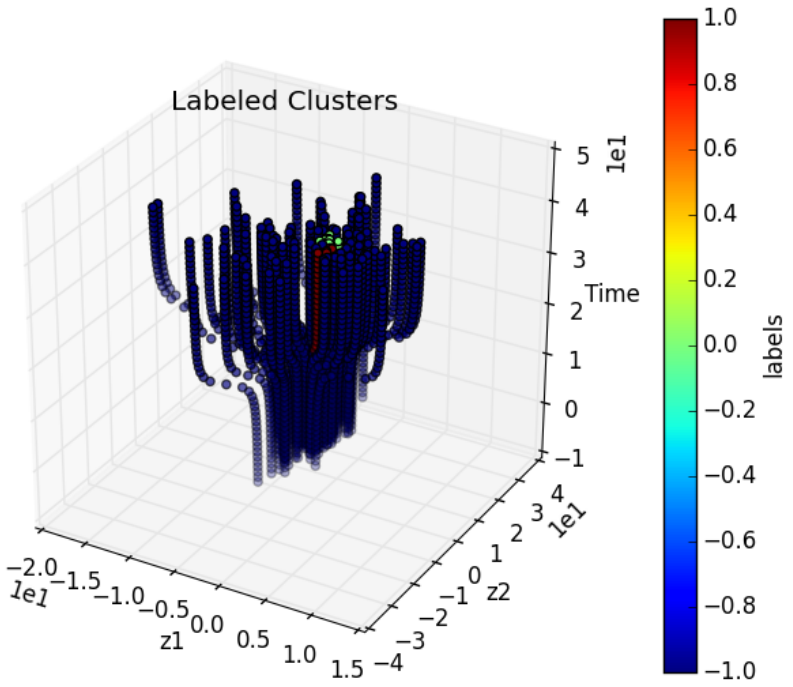


Figure 14. Clusters found with density-based spatial clustering of applications with noise.

## 4.3 Dimensionality Reduction

The dimensionality reduction portion of the TDM allows understanding of the presence of redundancy in the analyzed dataset. Especially in statistical analysis, it is important to remove information that is redundant with respect to explaining why the data are dispersed .A simple test of this uses the following model:

$$x_1[t] = x$$

$$x_2[t] = x + y\,t$$

$$x \sim U(0,100)$$

$$y \sim U(0,100)$$

In this model, there are two random variables, *x* and *y*. These random variables are independent and uniformly distributed between 0.0 and 100.0 (but the results would be similar with other distributions so long as the distributions are identical and independent). PCA dimensionality reduction was run for the two output variables $x_1$ and $x_2$.

As we can imagine in a $x_1$, $x_2$ two-dimensional plot, at t=0, all the points will be distributed exactly along the line $x_1 = x_2$, and therefore all the variance could be explained only by one principal component. With the time passing, the influence of the term "$+ y\,t$" grows and two components will be needed to explain the variance of the dataset.

As it can be seen from Figure 15, the explained variance by the first component is 100% at t=0 by the time passing the variables a $x_1$, and $x_2$ become less and less correlated. At the $\lim_{t\to\infty} x + y\,t = y\,t$ and therefore, $x_1$, and $x_2$ are completely uncorrelated. In this situation, two principal components will be

needed and each would explain only 50% of the variance. All this is well captured in the plot in Figure 15.

The vector components are constant as seen in Figure 16. Initially, the vector (√2/2,√2/2) explains 100% of the variance before the influence of the "+y t" term starts. As time increases, the second component vector (√2/2,-√2/2) is needed for the "+y t" term.

Results shown here were obtained using the ExactPCA option in RAVEN to determine the components and then measure the explained variance ratio.

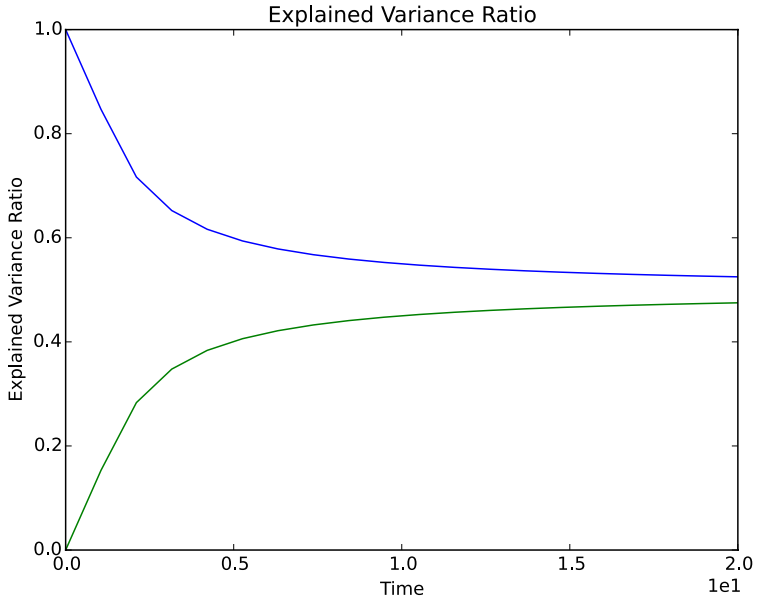


Figure 15. Explained variance ratio time evolution.

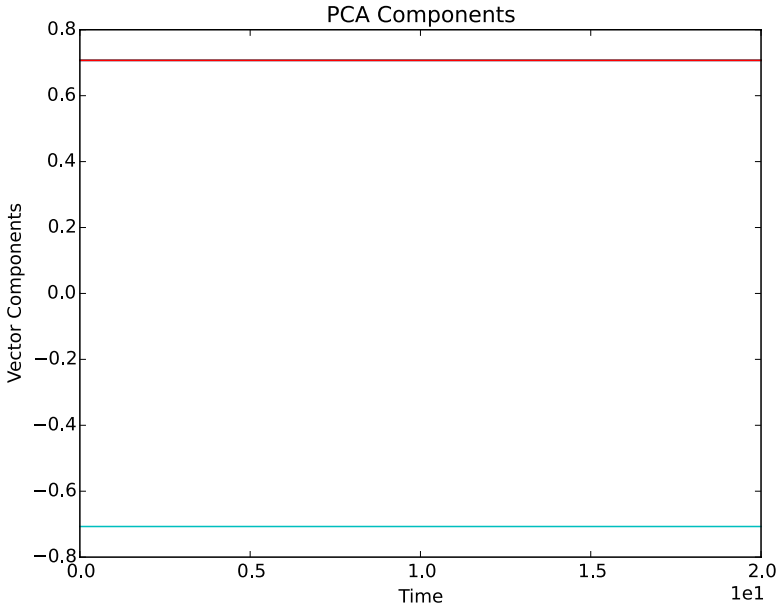


Figure 16. Vector components.

# 5. EXAMPLES OF USING TIME-DEPENDENT DATA MINING WITH BISON

For the TDM, a RAVEN input was created to use BISON and RAVEN and TDM. BISON [15,16] simulated a fuel element in a reactor with a power history as shown in Figure 17 and with a power spike highlighted in Figure 18. The BISON mesh used in the simulation is shown in Figure 19. The details of the experimental set up are provided in [17]. BISON was run 7000 times by RAVEN while varying 12 different parameters using uniform distributions and a Monte Carlo sampling strategy (Table 1) and 12 output parameters were monitored (Table 2). RAVEN ran 128 BISON simulations simultaneously with each using 16 MPI processes (total of 2048 cores simultaneously used).

The analysis reported below is only exemplificative of RAVEN new capabilities, a deeper interpretation of the data would be suggested as a follow-up of this activity to highlight better the relationship between the behaviors identified by the analysis reported here and the physics determining the detected behavior of the system. Defining better distributions and parameter ranges would be needed to make this case fully realistic, but at the same time, it is enough to show case an application of RAVEN TDM capabilities

Four sections follow, the first two, dispersion analysis, and relational analysis, are needed to verify the findings of the latter, which are clustering, and dimensionality reduction.

Table 1. Sampled input parameters and sampling bounds.

| Code Variable Name | Meaning | Unit | Lower Bound | Upper Bound |
|---|---|---|---|---|
| Fuel_thE | Fuel thermal expansion | 1/K | 8.5e-6 | 1.15e-5 |
| Clad_thE | Cladding thermal expansion | 1/K | 3.5e-6 | 6.5e-6 |
| Clad_thC | Cladding thermal conductivity | W/(m K) | 14.4 | 17.6 |
| Fuel_thC | Fuel thermal conductivity scaling factor | dimensionless | 0.9 | 1.1 |
| Grain_diffCoeff | Inter-granular diffusion coefficient | dimensionless | 0.1 | 10.0 |
| Power | Power scaling factor | dimensionless | 0.95 | 1.05 |
| Fuel_reloc | Fuel relocation scaling factor | dimensionless | 0.8 | 1.2 |
| Grain_radius | Grain radius scale factor | dimensionless | 0.4 | 1.6 |
| Clad_ox | Oxide scale thickness scaling factor | dimensionless | 0.6 | 1.4 |
| Clad_creeprate | Creep rate scaling factor | dimensionless | 0.7 | 1.3 |
| Gas_thC | Gas thermal conductivity scaling factor | dimensionless | 0.9 | 1.1 |
| Grain_res_param | Inter-granular resolution scale factor | dimensionless | 0.1 | 10.0 |

Table 2. Output monitored parameters.

| Code Variable Name | Meaning | Unit |
|---|---|---|
| midplane_hoop_stress | Midplane hoop stress on cladding | Pa |
| max_hoop_stress | Maximum hoop stress on cladding | Pa |
| max_vonmises_stress | Maximum von Mises stress on cladding | Pa |
| midplane_vonmises_stress | Midplane von Mises stress on cladding | Pa |
| max_clad_temp | Maximum temperature in the cladding | K |
| fis_gas_released | Released fission gas | moles |
| max_fuel_temp | Maximum fuel temperature | K |
| midplane_contact_pressure | Midplane contact pressure | Pa |
| avg_gap_conductance | Average gap conductance | W/(m K) |
| gas_volume | Interior gas volume | $m^3$ |
| ave_temp_interior | Average interior temperature | K |
| average_burnup | Average pellet burnup | MWd/kgU |

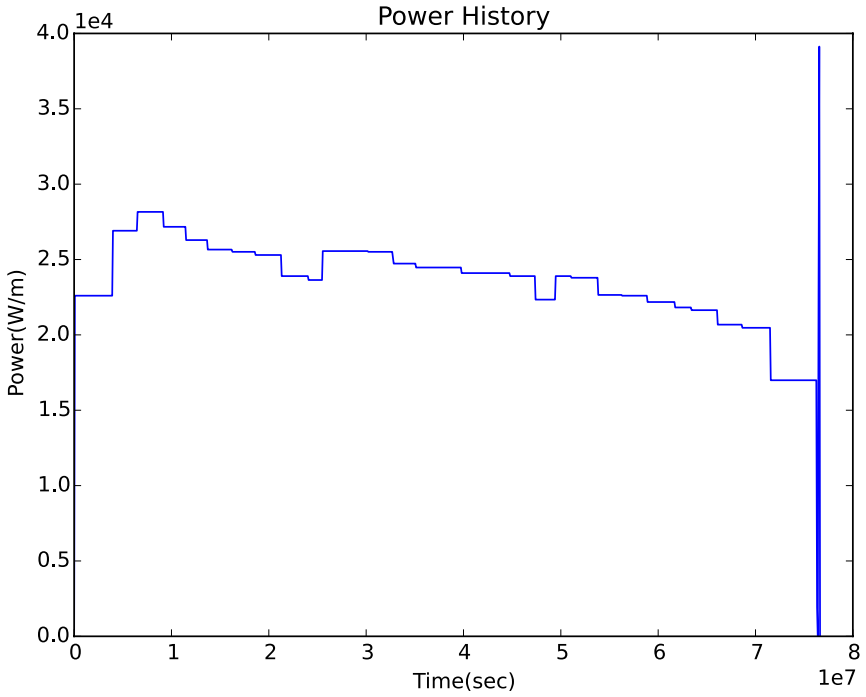


Figure 17. BISON power history used in the simulation.

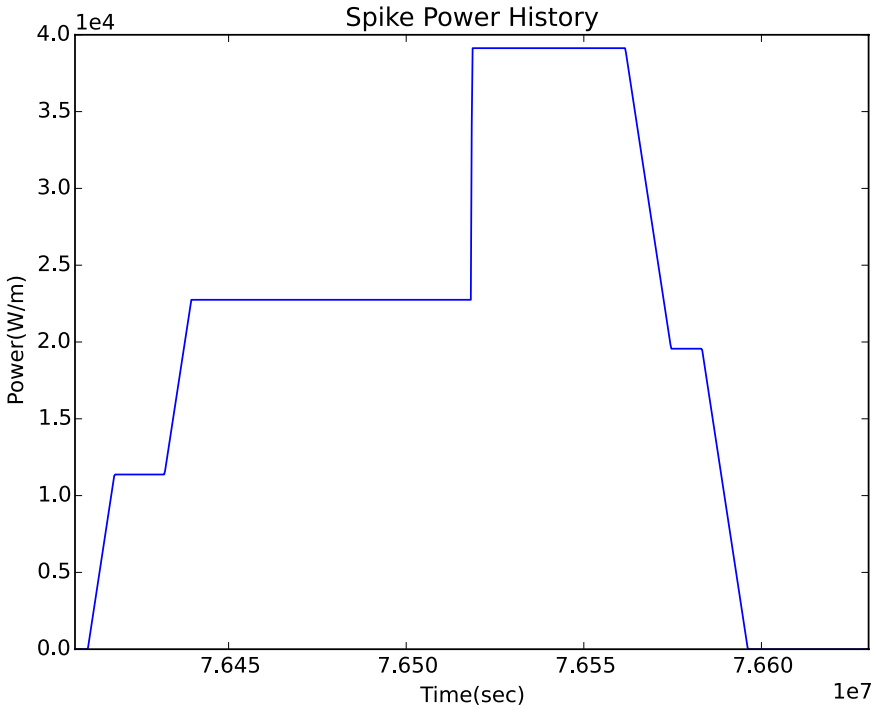


Figure 18. BISON simulation power spike.

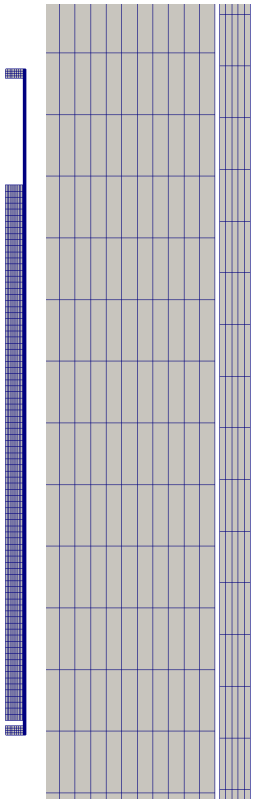

Figure 19. BISON mesh used in the simulation: smeared pellet stack with hafnium insulator end pellets.

## 5.1 System Response Dispersion Analysis

In the following sections, the dispersion of the BISON runs is analyzed. The example chosen parameter is the maximum hoop stress (Figure 20 and Figure 21). The maximum hoop stress has been analyzed using both time and burnup as a pivot variable. It is interesting to note how the use of burnup as a pivot variable (any of the monotonically increasing output variables can be used as pivot variable in RAVEN instead of the time), it compresses not relevant part of the experiment and expand the plots in correspondence of the region of interesting transient in the fuel behavior. The plots are, in fact, more readable when the system evolution is plot versus the burnup. The plots present the mean, the median (average value of the minimum and maximum value detected), 5% percentile (lower value encompassing only 5% of the possible outcomes), and 95% percentile (higher value encompassing 95% of the possible outcomes).

The reason to report here these post processed values of the some of the output figure of merits is to match some of the finding from those analyses with the finding from the advanced data mining. For example one important point, clearly, is around ~3.3 1.e7 sec or 1.75 MWd/kgU, were hoop stress initiate the inversion of sign (from negative to positive). As we will see later on, this point coincide with a change in the behavior of the system detected both from clustering and dimensionality reduction. The change in the system behavior is most likely due to the beginning of the clad pellet interaction.

An example of an interesting effect that could need further investigation is in correspondence of a burnup of ~0.02 MWd/kgU. It clearly looks like there is a mechanism of release of the hoop stress that is not present in the whole spectrum of the sampled input parameters. This is highlighted by the fact that the 95% percentile curve is steadily growing while the 5% percentile has a sharp drop. The opposite situation appears at around a burnup of 0.027. It seems, therefore, that there is a set of input parameters that are capable to delay the release of the stress.

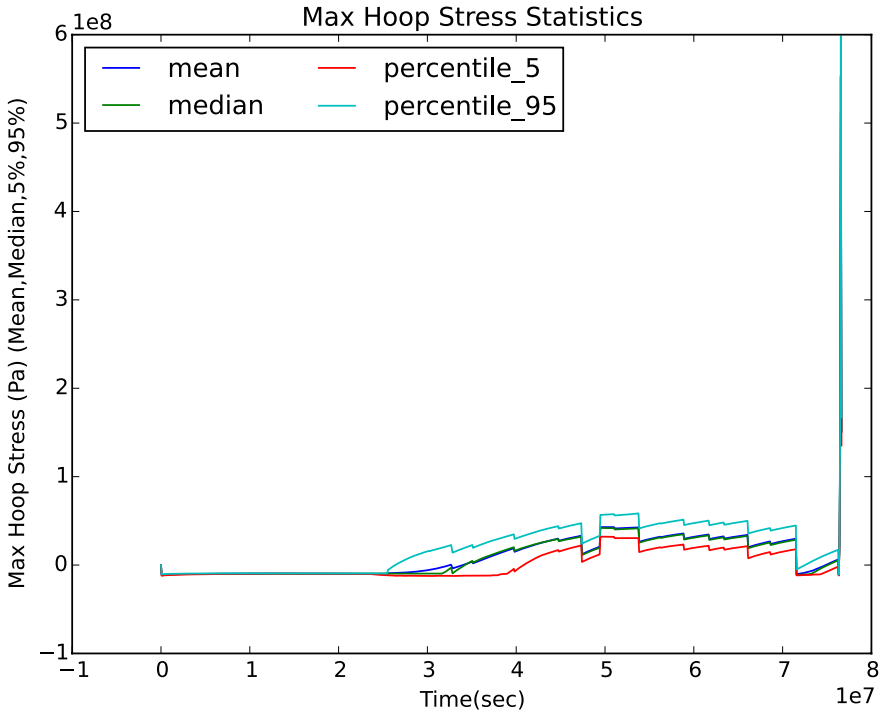


Figure 20. Maximum hoop stress with respect to time.

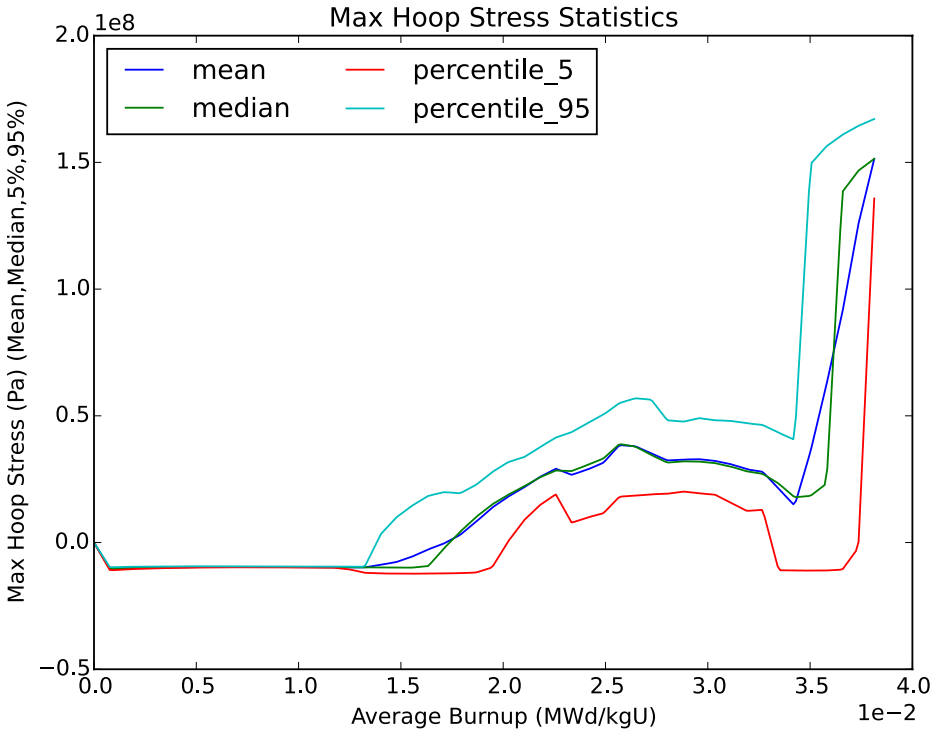


Figure 21. Maximum hoop stress with respect to burnup.

## 5.2 Relational Analysis

For the relational analysis we present first the time evolution of the covariance of the fission gas release during the transient except the final ramp (Figure 22) and then the final ramp analysis in a separate plot: Figure 23. Also the correlations for the first part of the transient and for the final spike for the maximum hoop stress are shown respectively in Figure 24 and Figure 25.

The covariance analysis is used to determine which components are most influential toward the dispersion of the code output. RAVEN can compute the covariance (and also the correlation) for a set of variables, which can be a mixture of input output, only output, or only input. Depending on the set of variables addressed by the analysis, the covariance and correlation can be used, therefore, as supervised or unsupervised learning algorithm. Since the unsupervised learning approach, in the output space, would lead to result comparable with the ones that will be obtained by the PCA, here we show the supervised approach where the relationship between input and output is studied.

In a time dependent relational analysis for data mining the most relevant part is not the values of the contribution to the dispersion but rather changes and trends. As we can see, the grain diffusion coefficient is the largest positive contributor; among the other positive contributors we have the inter-granular resolution scale factor, and power. The transition spots identified by the dispersion analysis due to the clad fuel contact appear clearly at around ~3.3 1.e7 sec in the maximum hoop stress plots. The larges negative contributor is the grain radius followed by the thermal conductivity of the clad and fuel. Clearly we would expect this trend to be confirmed by the advanced data mining algorithms, in particular by the cluster separation analysis.

In fact, a type of analysis, which is important in terms of deciding the visualization projection of the clustering approach, is the correlation analysis. The correlation, in practical terms, is a sensitivity analysis and therefore we should be using this analysis to guide the visualization by deciding which ones are the most important variables when trying to visualized the dispersion of the clusters in the input space.

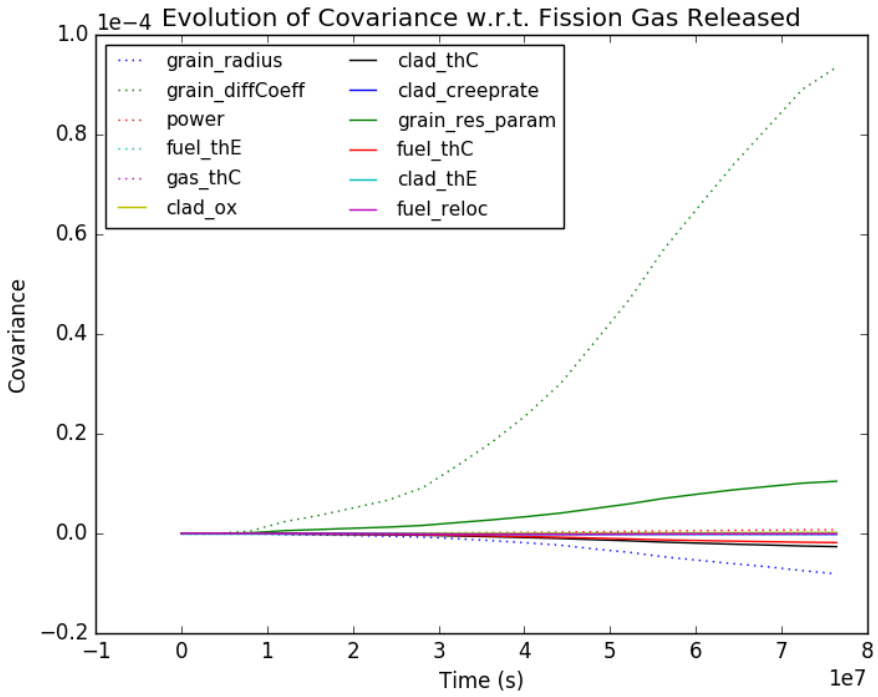


Figure 22. Covariance evolution for the fission gas release excluding final ramp.

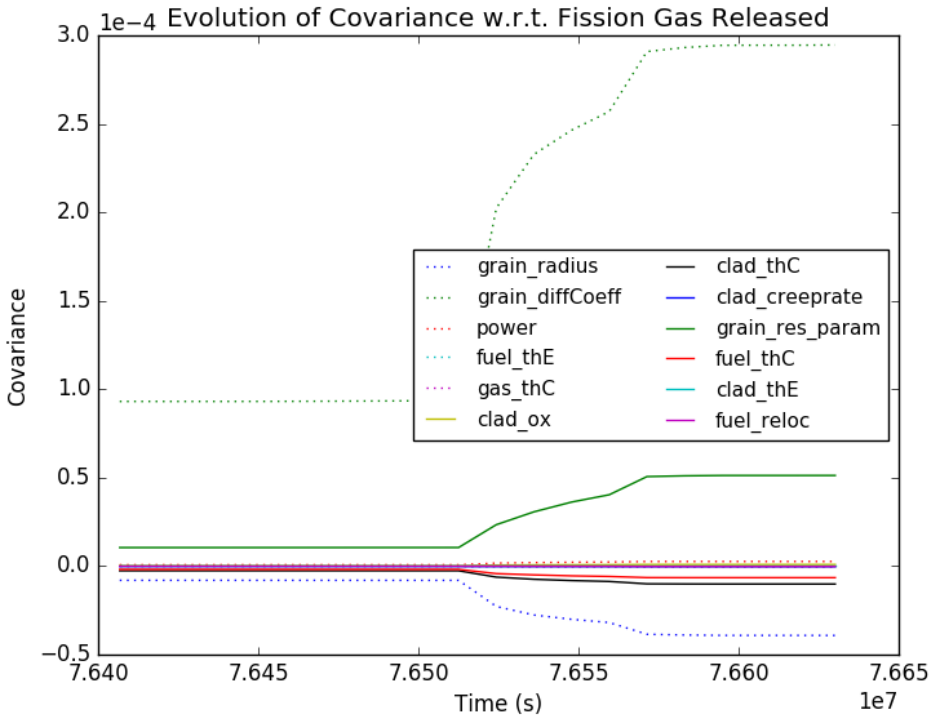


Figure 23. Covariance evolution for the fission gas release in the final ramp.

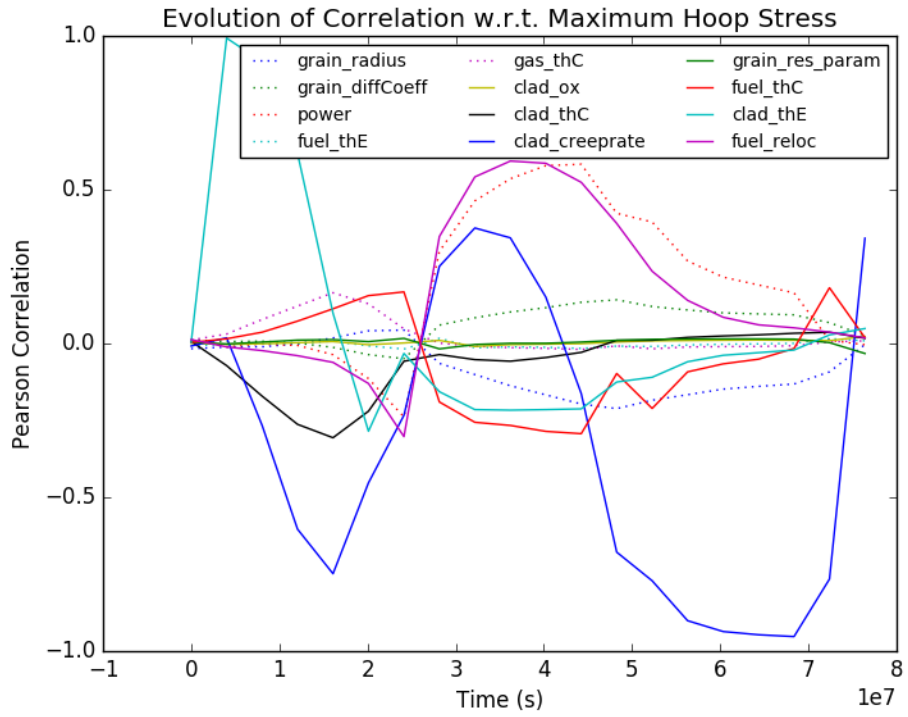


Figure 24. Correlation (Pearson moments) evolution for the fission gas release excluding final ramp.

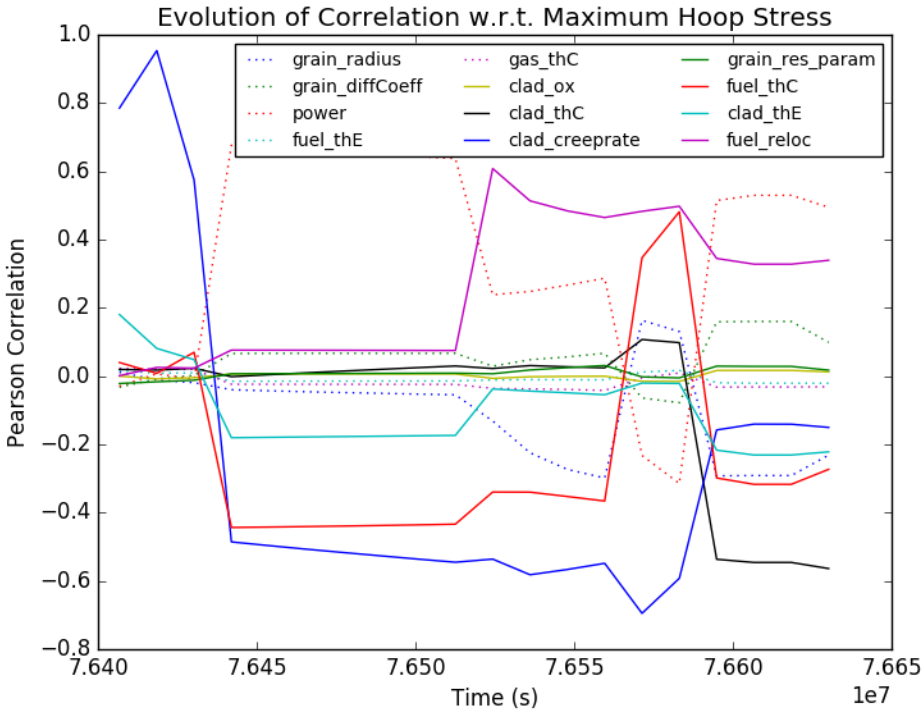


Figure 25. Correlation (Pearson moments) evolution for the fission gas release in the final ramp.

## 5.3 Clustering

The next set of examples is focused on the illustration of the clustering capability. Colors in the plots are coherent in the whole section with respect to the clustering labels. The results shown in the following sections are the outcome of the application of K-means based clustering. Three clusters were requested from the algorithm. The clustering was performed with respect to all the output variables. The clustering has been applied using both time and burnup as pivot variables. For brevity, we just show the plots of the centroid trajectory with respect to the mid-plane von Mises stress on cladding and the average interior temperature.

Clearly the average interior temperature shows a little degree of separation between the clusters even in the final power ramp. This could be appreciated in Figure 26 and Figure 27. This means that the average interior temperature has a very low dispersion not being a good discriminant for the determination of the clusters. In other words, the average interior temperature is not strongly impacted by the variation of the input parameters considered within their ranges of variations.

As it could be inferred from Figure 28, the mid-plane von Mises stress is a much stronger discriminant for the clustering algorithm, indicating a large sensitivity of this figure of merits with respect to the variable sampled and the ranges chosen. At the same time von Mises stress seems to lose its value as cluster discriminant when the contact (as indicated by the initial dispersion analysis) between the fuel at the cladding happens (~3e7 sec). Figure 29 shows a very complex trajectory of the centroids during the final power ramp. The sudden drop of the stress for the blue centroid most likely is simply due to the fact that this cluster seems to have a slower response time to the power ramp and therefore there is enough time spent at low stress level to capture it. The other clusters simply spent too little time at low stress that the time resolution has not captured it (RAVEN can perform a sub sampling of the time step). This could be confirmed looking at the cluster trajectories with respect the Figure 30 and Figure 31 that show clearly

a cluster with a large delay in time to respond to the power ramp. These are currently a qualitative consideration that would need more investigations, especially to exclude the presence of outliers that might mislead the clustering algorithm.

There are four more plots following; these plots are made seeking to identify the discriminant variables with respect the generation of the cluster locations. We have to recall that, in the time dependent approach implemented, the set of the realization in the input space which belongs to a cluster, they change over time since the clustering is done at each time step (possibly accounting for some inertia like terms).

Figure 32 shows the input space realization colored depending the cluster belonging at t=5e7. As it could be inferred, from Figure 28 This is a time region where the clusters (at least with respect the von Mises stress) are quite separate. Consequently, in this time region, when choosing a good set of projection parameters, the cluster shows a good degree of separation. Of the three parameters chosen (power, fuel conductivity, and grain radius) the power and the fuel conductivity are the dominant label discriminant. This is in reasonable agreement with the relational analysis performed (on a closely correlated variable, which is the max hoop stress). The proper procedure would have been to perform a covariance analysis (using the supervised learning approach) for all the outputs (which are the clustering space) with respect the input space and use the most significant direction for the 3D projection. This analysis is just demonstrative of the newly developed capabilities.

Figure 33, instead is built with the opposite goal. A bad choice in the 3D projection variables, it makes the clustering in the input space not readable anymore. It is interesting to observe that the creep rate in spite of a big ranking obtained in the relational analysis does not seems to be an explanatory value for clustering. Once more, this is, most likely, due to the incompleteness of our relational analysis as a guide for choosing the projection space (all sensitivity analysis for all variables should have been done).

Figure 34 and Figure 35 focuses instead in the region of the final ramp. While the behavior of the input space variables with respect the capability of being discriminant of the clustering classification is the same as before, the third cluster is almost non-present. Figure 34 shows a very limited number of red realizations. This seems to indicate that the cluster is most likely the results of numerical artifacts and the point that they belong to the cluster outliers. This statement, to be confirmed, would require to manually identifying those runs to verify possible anomalies. Clear this work is outside of the scope of this report but still shows the power of the data mining approach.

Figure 37 Figure 38 and Figure 38 illustrate a different way to locate the originating combination of the input parameters for each cluster (the time of the clustering time snapshot chosen for this analysis is t=5e7s). The capability of an input parameter to be important with respect to the classification of a point with respect the cluster generated is summarized in the following Table 3. The importance with respect to the classification in the different clustering is derived by the distortion from a uniform distribution of the histograms. This would be more complex if the sampling distributions were not uniform, since this would have biased the density distribution of the sampling and therefore affecting the histograms. In this case the distortion from the expected histogram should have been considered.

It is valuable to reiterate that this importance ranking is only for the specific condition of the fuel at t=5e7s and plus it represent the relevance with respect to the clustering labeling. To understand how the importance with respect to the labeling transfers to the importance with respect to the behavior of the fuel, we need to look to the projection of the centroids in the different dimensions of the output space and see in which dimensions the distance between the centroid is maximum. Those output dimensions are therefore the most impacted by the input parameters that are the stronger discriminating factors for the cluster labeling.

For example this type of analysis indicates that the high importance variables in the table below are highly impactful with respect to the middle plane Von Mises stress and not with respect to the average

interior temperature at t=5e7s. This can be seen by looking at the centroid separation in Figure 28 and Figure 26.

Table 3. Sampled input parameters importance with respect cluster separation at t=5e7s.

| Code Variable Name | Meaning | Unit | Importance |
|---|---|---|---|
| Fuel_thE | Fuel thermal expansion | 1/K | Low |
| Clad_thE | Cladding thermal expansion | 1/K | Low |
| Clad_thC | Cladding thermal conductivity | W/(m K) | Medium |
| Fuel_thC | Fuel thermal conductivity scaling factor | dimensionless | High |
| Grain_diffCoeff | Inter-granular diffusion coefficient | dimensionless | Medium |
| Power | Power scaling factor | dimensionless | High |
| Fuel_reloc | Fuel relocation scaling factor | dimensionless | Low |
| Grain_radius | Grain radius scale factor | dimensionless | Medium |
| Clad_ox | Oxide scale thickness scaling factor | dimensionless | Low |
| Clad_creeprate | Creep rate scaling factor | dimensionless | Low |
| Gas_thC | Gas thermal conductivity scaling factor | dimensionless | Low |
| Grain_res_param | Inter-granular resolution scale factor | dimensionless | Low |

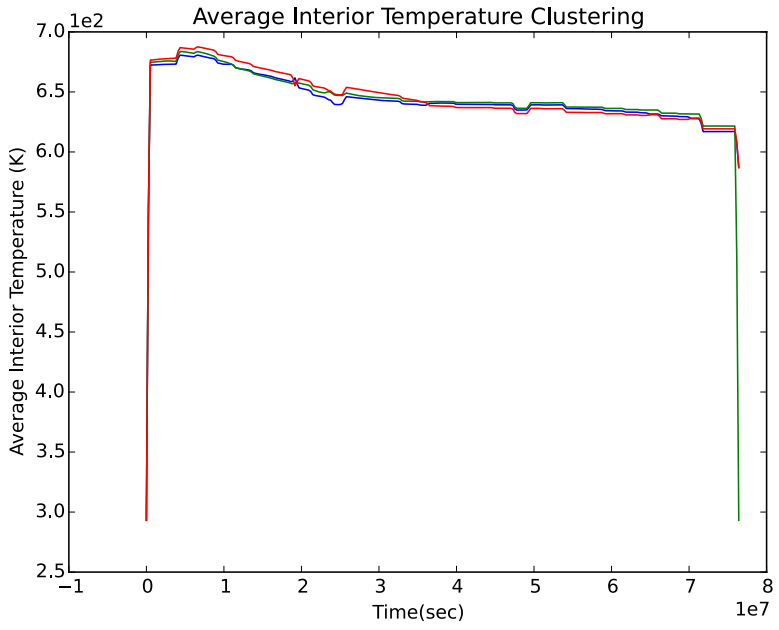


Figure 26. Cluster centroids trajectory over time as a function of the average interior temperature.

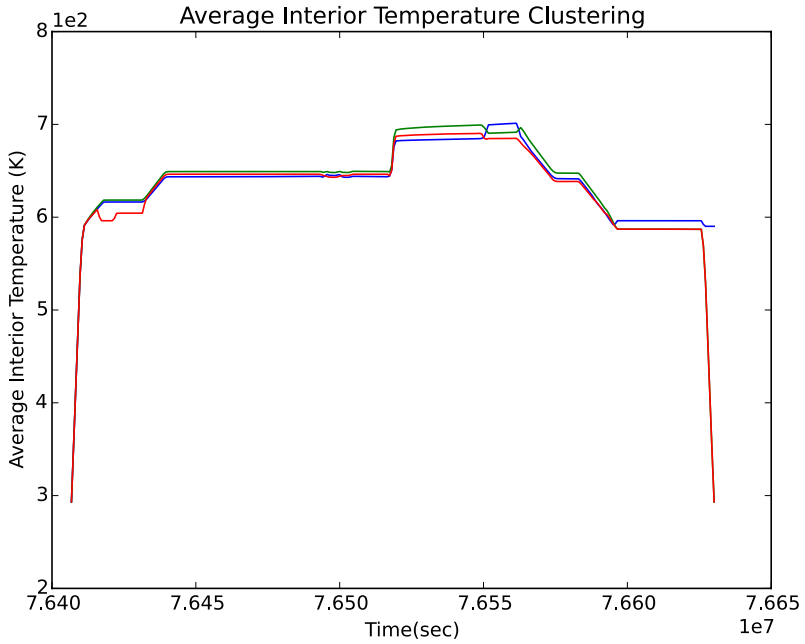


Figure 27. Cluster centroids trajectory over time as a function of the average interior temperature (detail of the final ramp).

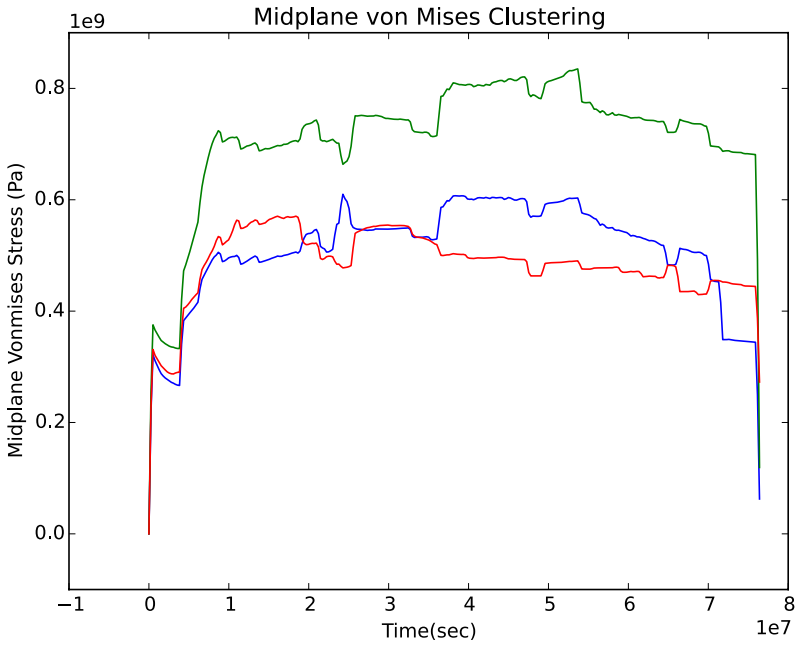


Figure 28. Cluster centroids trajectory over time as a function of the middle plane von Mises stress.

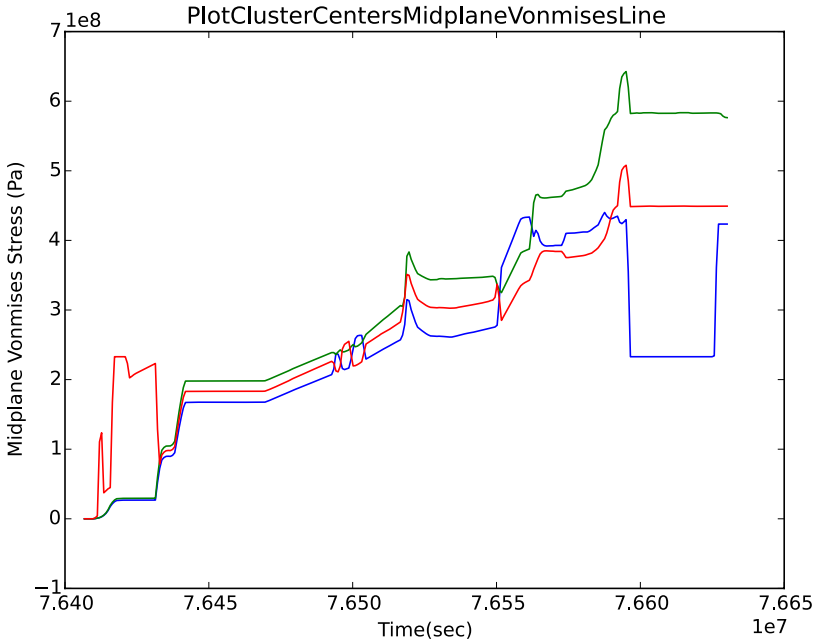


Figure 29. Cluster centroids trajectory over time as a function of the middle plane von Mises stress (detail of the final ramp).


Figure 30. Cluster centroids trajectory over burnup as a function of the middle plane von Mises stress.

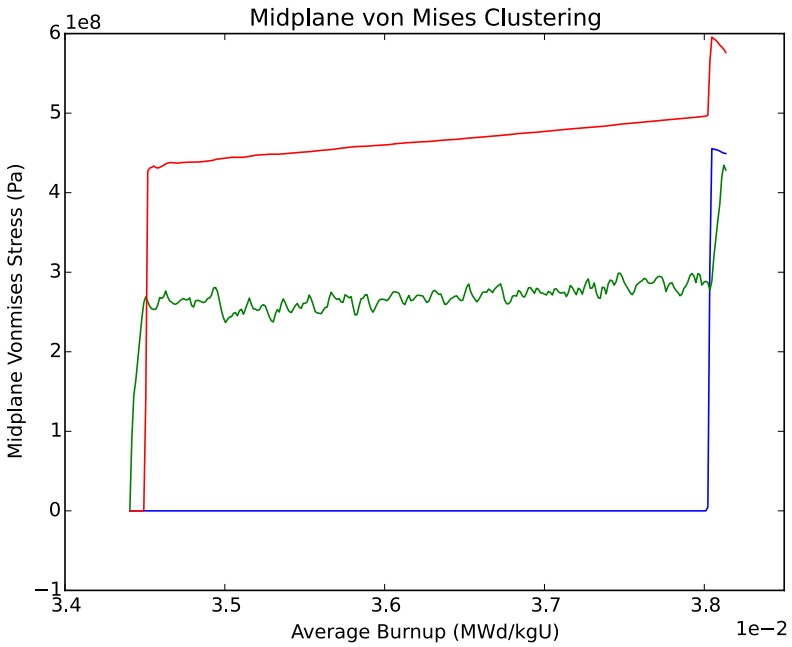


Figure 31. Cluster centroids trajectory over burnup as a function of the middle plane von Mises stress (final ramp detail).

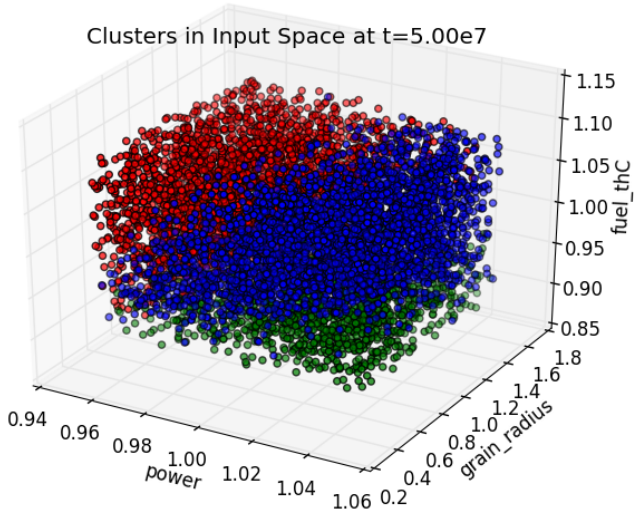


Figure 32. Input space realization colored depending their cluster belonging at t=5e7s.

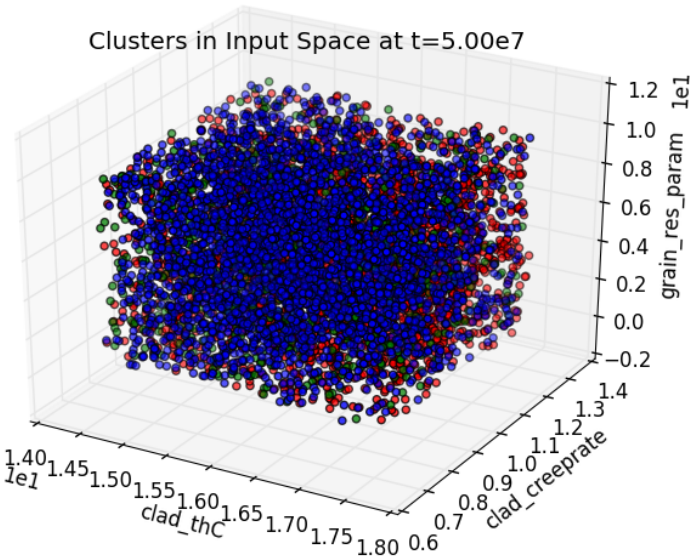


Figure 33. Input space realization colored depending their cluster belonging at t=5e7s.

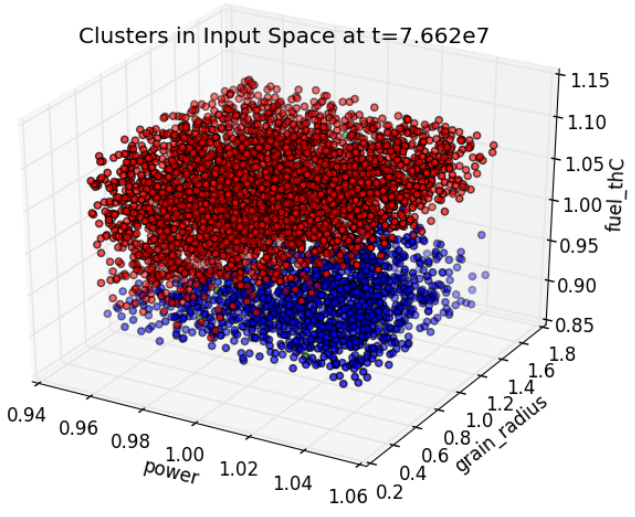


Figure 34. . Input space realization colored depending their cluster belonging at t=7.662e7s.

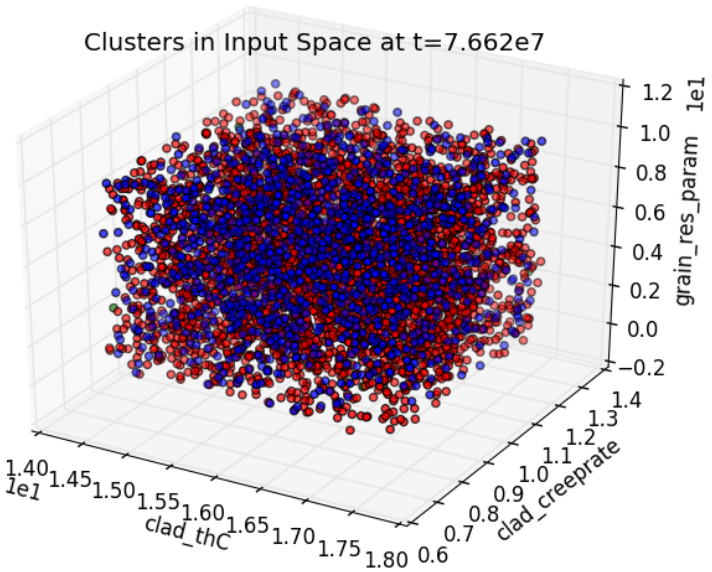


Figure 35. Input space realization colored depending their cluster belonging at t=7.662e7s.

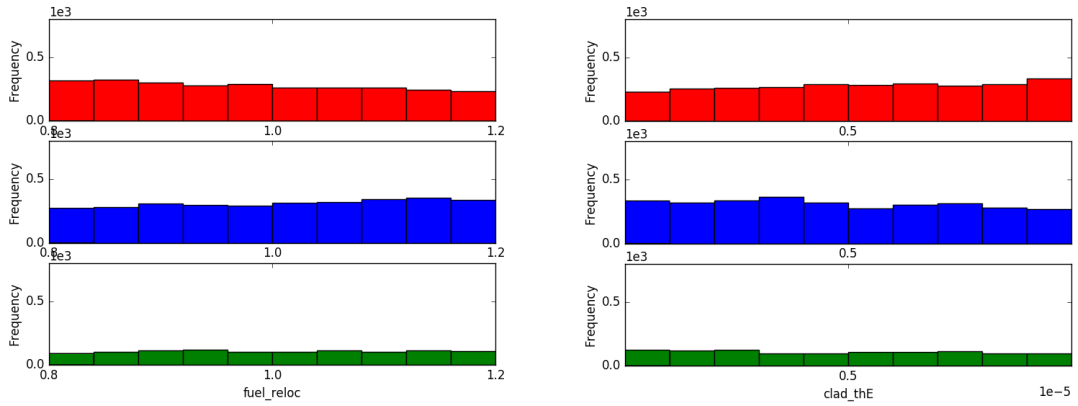


Figure 36. Histogram of the originating points in the input space of the three clusters for four input variables (t=5e7s).

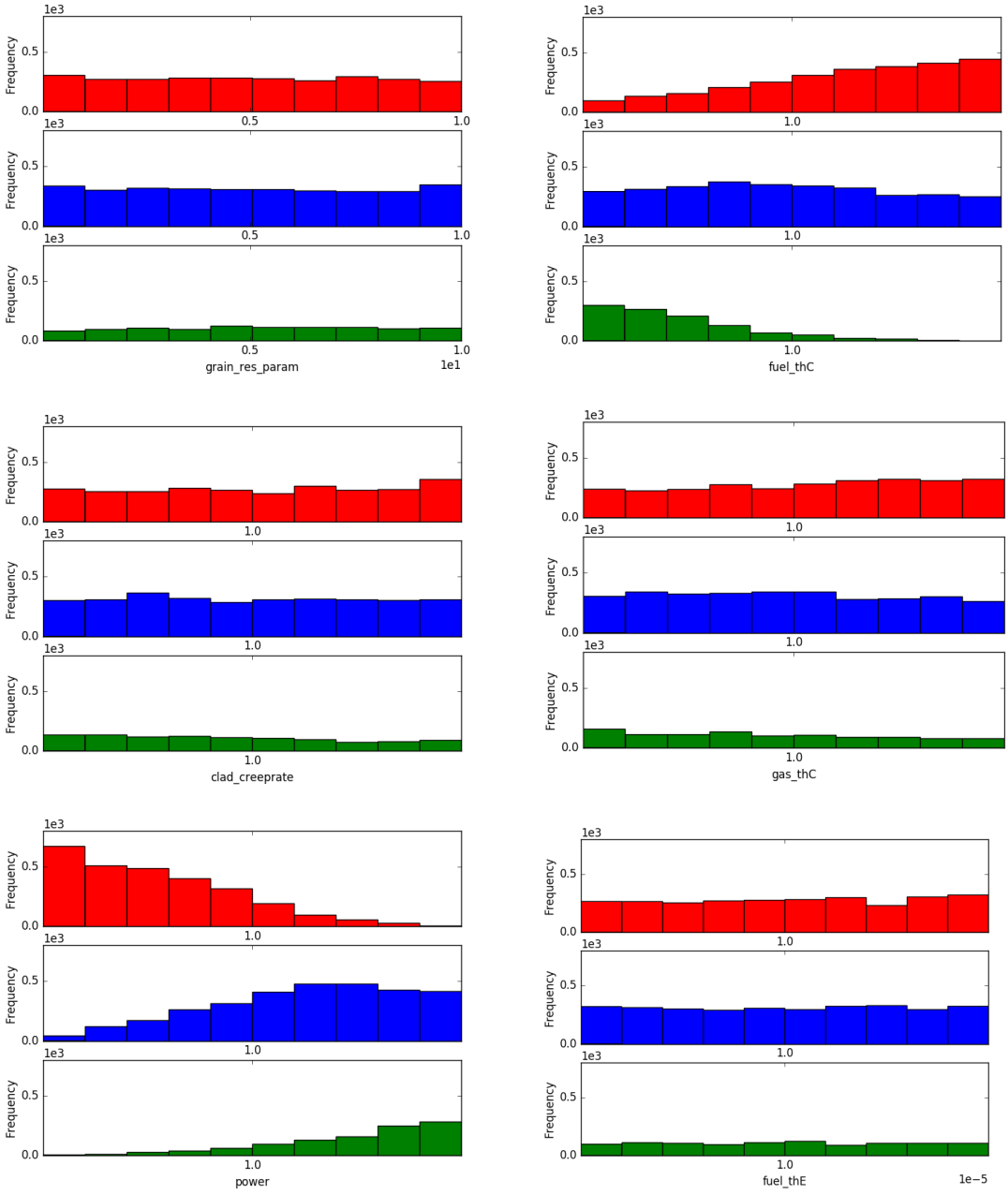


Figure 37. Histogram of the originating points in the input space of the three clusters for six input variables (t=5e7s).

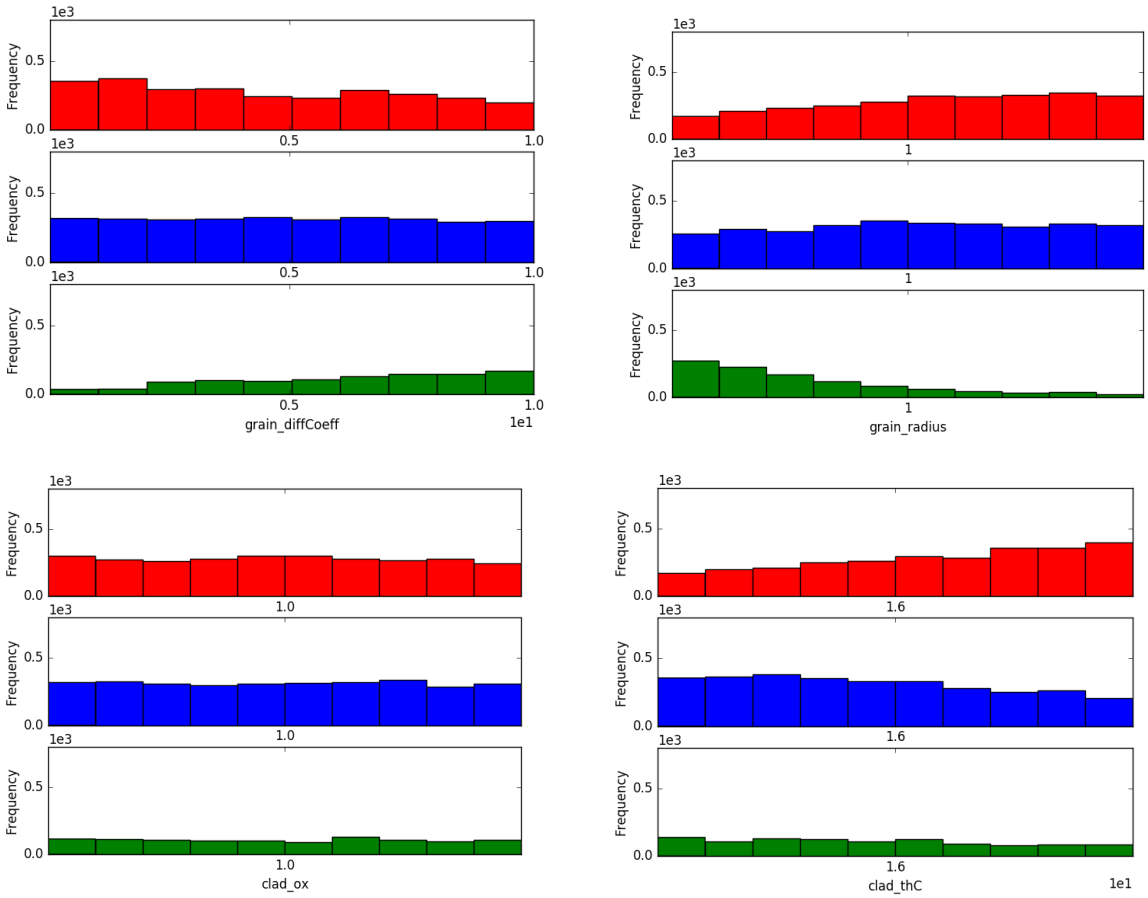


Figure 38. Histogram of the originating points in the input space of the three clusters for four input variables (t=5e7s).

## 5.4 Dimensionality Reduction

The results of the dimensionality reduction show different behavior along the power history. The amount of the explained variance varies over the course of time as seen in Figure 39 through Figure 42. It is important to notice how only few components (~3) are capable to explain most of the variance, indicating a high degree of correlation among the considered outputs. For example, the plots of the first component before the final ramp (Figure 43) indicate that the outputs gas volume and midplane hoop stress are strongly positively correlated, while max fuel temperature and fission gas released are both negatively correlated. It is also relevant to notice that the average gap conductance switches from being negatively correlated to positively correlated in correspondence of the contact of the pellet with the clad. Opposite behavior is shown by max hoop stress and interior temperature. In Figure 45 (final power ramp) the inversion of the correlation is not anymore presented since contact has already taken place.

The macroscopic changes in the physical behavior of the system (identified first in the dispersion analysis and confirmed by the relational analysis) not only are detected by the trend analysis of the components of the principal components, but also by changes of the ratio of explained variance along the transient.

Figure 44, and Figure 46 perform the same type of analysis for the third component. The third component accounts for phenomena that are less macroscopic and of lesser impact in determining the overall response of the system. The amplitude of the components is smaller, the fluctuations are much

more frequents, most likely they overlap with the small ramp that characterize the microstructure of the power history. An overlap plot with the mean values and the power history could be used to investigate this possibility.

This type of information might be used in designing an experiment when deciding which quantities should be monitored. Redundant quantities could be trimmed saving money in the experimental sensors. Moreover, when the explained variance changes sensibly, it indicates a change in the representative physics dominating the problem. This, therefore, provides information concerning where a sensitivity analysis could be used to identify changes in the relevant inputs following the importance of the different physical phenomena.

At this point there is no claim on the quantitative accuracy of the prediction made with the data mining tools, which generally speaking are qualitative in nature. It would be necessarily to use more time and a cross cutting team, including a fuel performance analyst and an expert of the methodologies so far illustrated to investigate and explain any possible agreement/discrepancies with respect to the expected physical interpretation of the experiment.

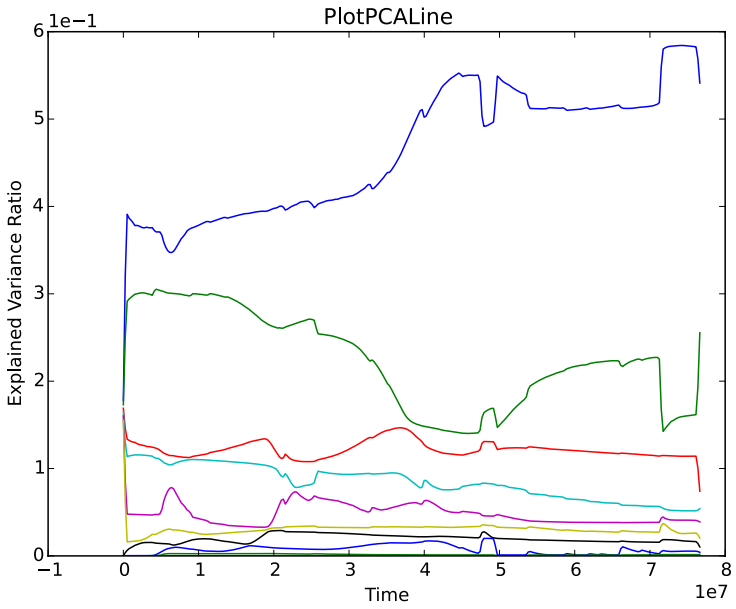


Figure 39. Explained variance ratio for each component with respect to time.

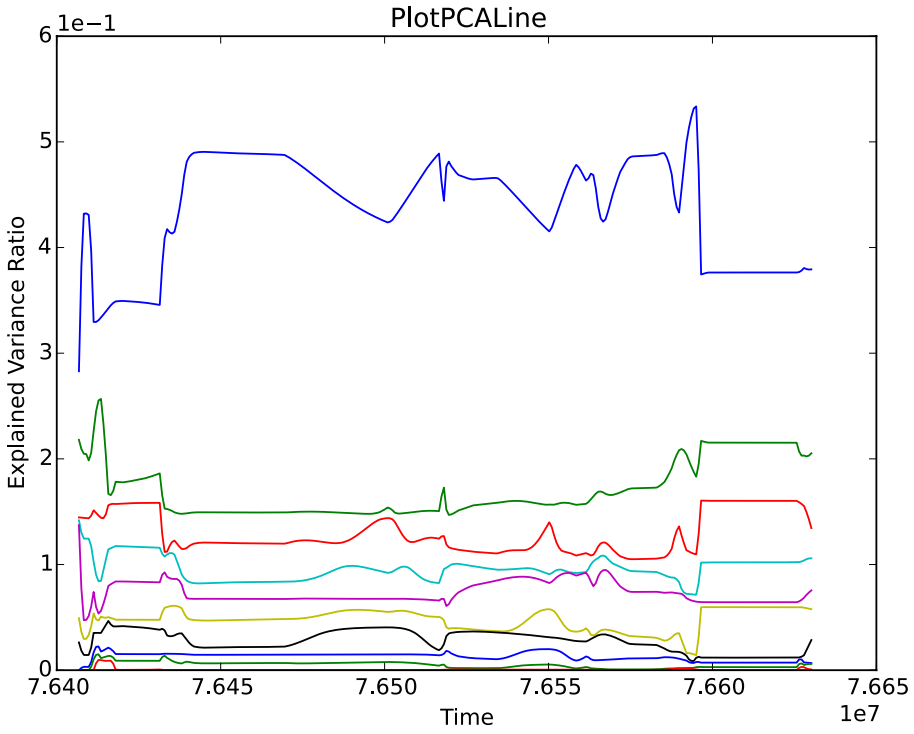


Figure 40. Explained variance ratio for each component with respect to time (final power ramp detail).

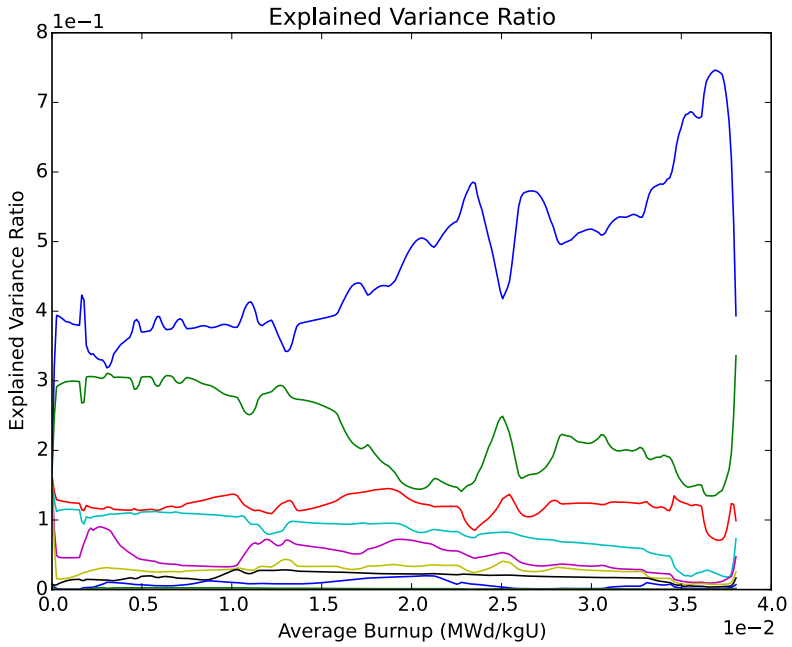


Figure 41. Explained variance ratio for each component with respect to burnup.

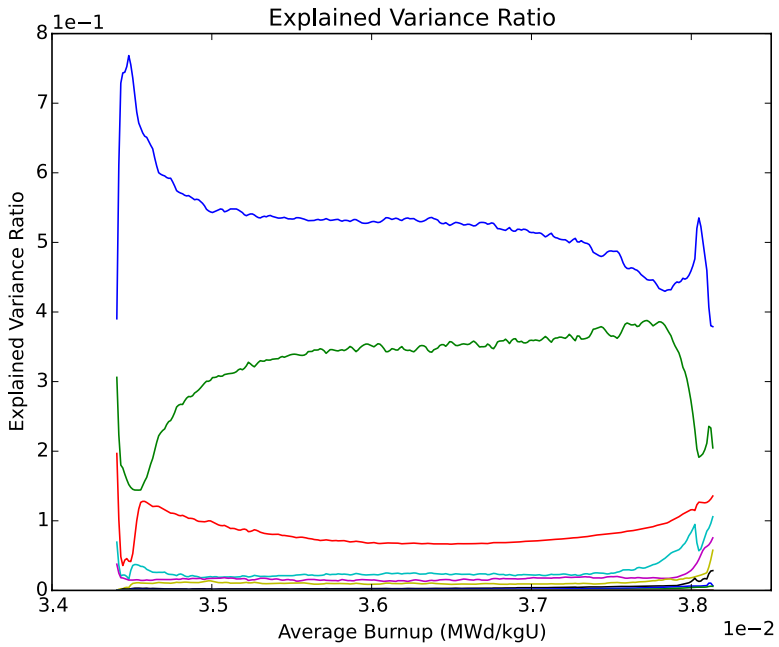


Figure 42. Explained variance ratio of each component with respect to burnup (final power ramp detail).

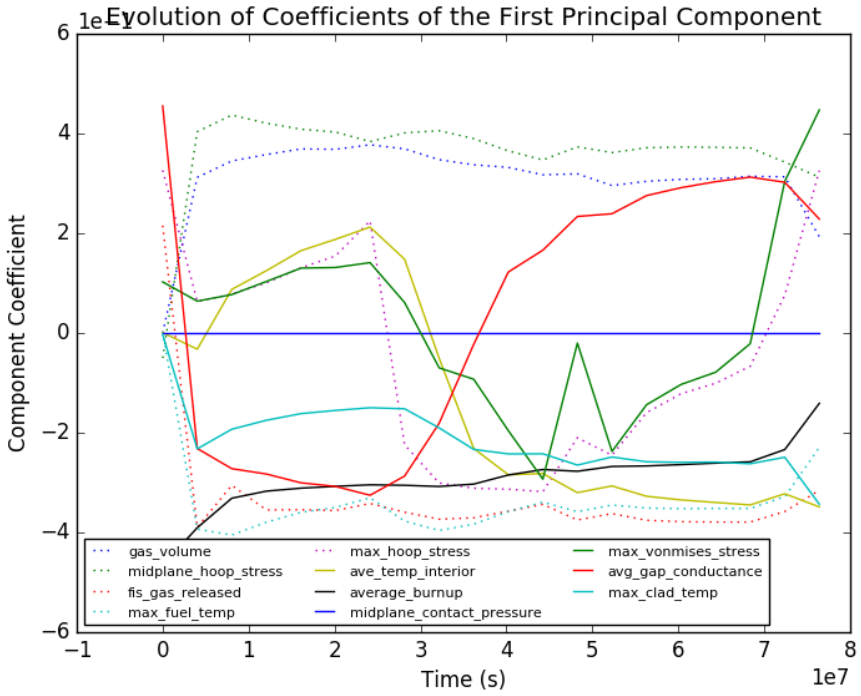


Figure 43. First principal components during the transient excluding the final ramp.

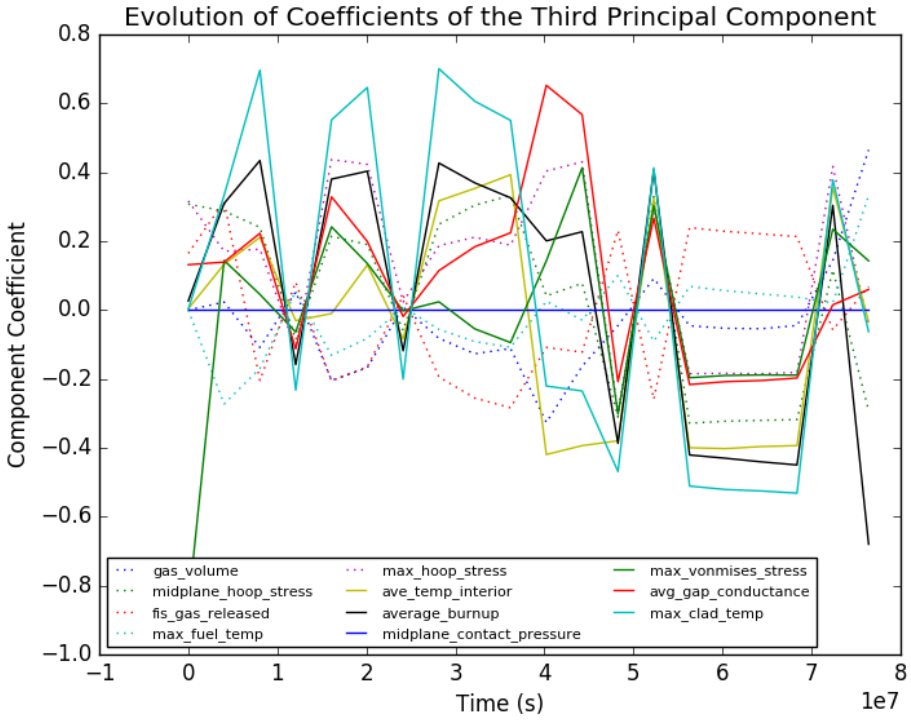


Figure 44. Third principal components during the transient excluding the final ramp.

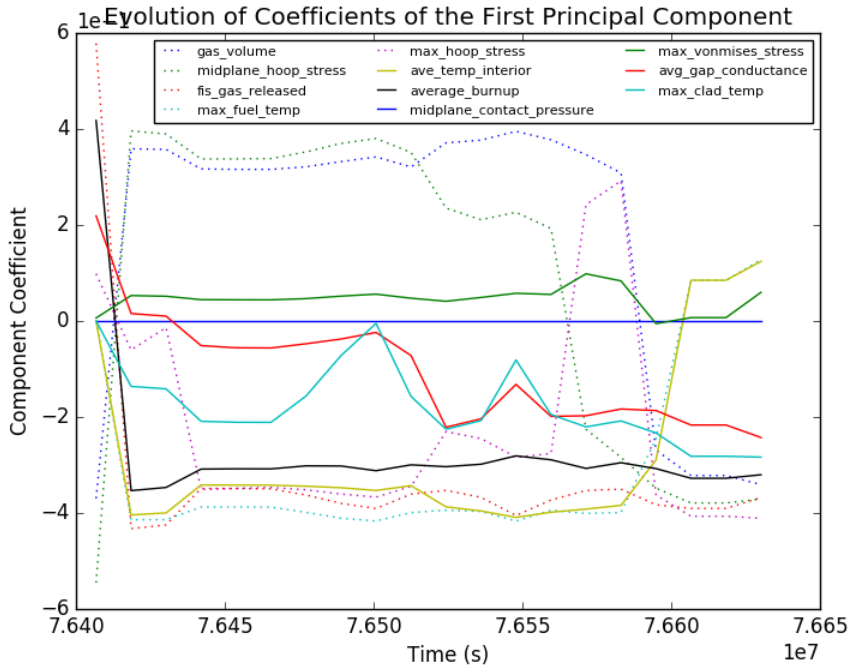


Figure 45. First principal components during the transient during final power ramp detail.

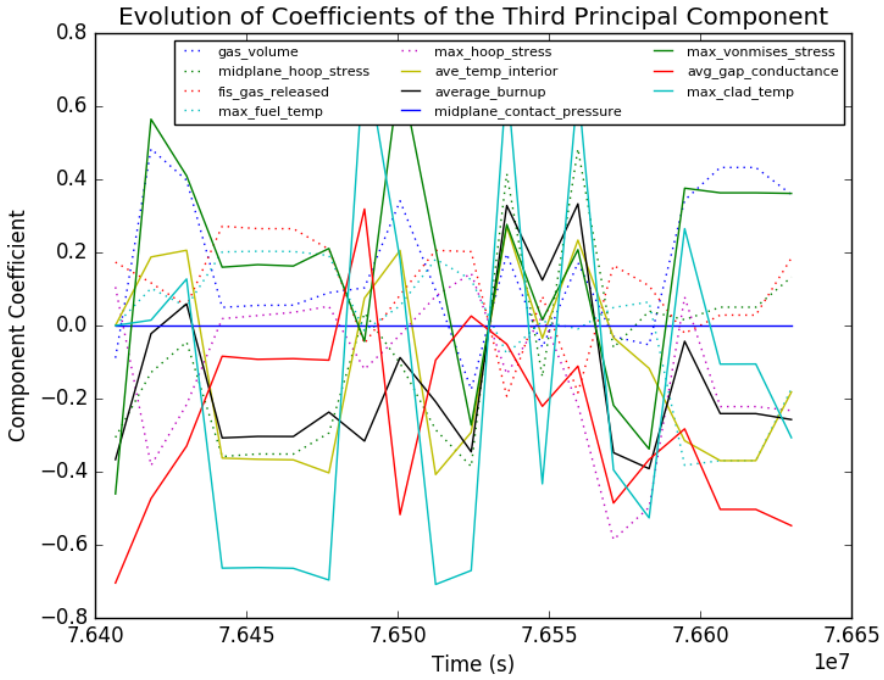


Figure 46. Third principal components during the transient during final power ramp detail.

# 6. CONCLUSION

TDM has been added to RAVEN. The implementation has created a general infrastructure that allows data mining analysis to be extended over time. A recall of the theory of the most used of those post-processing capabilities, with particular emphasis on clustering and dimensionality reduction, has been provided in Section 2. The static post-processing capabilities have been extended to the time-dependent domain by means of the theory presented in Section 3. The implementation of the TDM capabilities have been tested and illustrated in three examples presented in Section 4. Finally, Section 5 presents the application of the TDM to a realistic case based on use of the BISON code for the simulation of an experiment.

As presented in Appendixes A and B, the relative entries in the RAVEN User Manual have been updated to reflect the new capabilities and 20 new regression tests have been created.

The new features implemented show capabilities for better understanding the physics underlying the overall response of modeled systems, which could have implication in helping engineers to better design new systems, experiments.

The clustering could be used to highlight which dimension presents the highest dispersion and how the different physics of the systems alternate in determining the response of the system.

The dimensionality reduction could be used to identify redundant output signals, correlation among output variables, and the real dimensionality of the problem. Once more, the changes in the load and explained variance of the outputs over time signal changes in the system behavior that could provide important clues on the physical mechanisms dominant the system response.

At this point any conclusion drawn on the BISON simulation is clearly demonstrative and a close cooperation with some fuel experts, probably will be necessary for a correct interpretation of all data and analysis generated.

# 7. ACKNOWLEDGEMENTS

We thank Giovanni Pastore of INL and Davide Pizzocri and Tommaso Barani of Politecnico of Milan, Italy (working as interns at INL) that supplied the BISON input file used and with assistance interpreting the BISON results.

# 8. REFERENCES


1. Sen, Sonat, Daniel Maljovec, Andrea Alfonsi, and Cristian Rabiti, *Developing and Implementing the Data Mining Algorithms in RAVEN*, INL/EXT-15-36632, September 2015.
2. Cristian Rabiti, Andrea Alfonsi, Joshua Cogliati, Diego Mandelli, Robert Kinoshita, Sonat Sen, Congjian Wang, Jun Chen, *RAVEN User Manual*, Technical Report, INL/EXT-15-34123, February 2016
3. Xu, Rui, and Donald Wunsch II, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, Vol. 16, No. 3, pp. 645–678, May 2005.
4. Mandelli, D., A. Yilmaz, T. Aldemir, K. Metzroth, and R. Denning, "Scenario clustering and dynamic probabilistic risk assessment," *Reliability Engineering & System Safety*, Vol. 115, pp. 146–160, July 2013.
5. Mendelson, B., *Introduction to Topology*, New York: Dover Publications, 1990.
6. Pedregosa, F. et al*., Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research, Vol. 12, pp. 2825–2830, 2011, http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html.
7. Dueck, Delbert, *Affinity Propagation: Clustering Data by Passing Messages*, Dissertation: University of Toronto, Toronto, Ontario, Canada, 2009.
8. Wu, Kuo-Lung, and Miin-Shen Yang, "Mean shift-based clustering," *Pattern Recognition*, Vol. 40, No. 11, pp. 3035–3052, November 2007.
9. Comaniciu, D., and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 5, pp. 603–619, May 2002.
10. Russell, Stuart, and Peter Norvig, *Artificial Intelligence, A Modern Approach*, 3rd Edition. Pearson Education Limited, 2014.
11. Görür, Dilan, and Carl Edward Rasmussen, "Dirichlet process Gaussian mixture models: Choice of the base distribution." *Journal of Computer Science and Technology*, Vol. 25, No. 4, pp. 615–26, July 2010, DOI 10.1007/s11390-010-1051-1.
12. Blei, David M., and Michael I. Jordan, "Variational inference for Dirichlet process mixtures," *Bayesian Analysis*, Vol. 1, No. 1, pp. 121–144, 2006.
13. Bishop, Christopher M., "Pattern recognition," *Machine Learning*, Vol. 128, 2006.
14. Pearson, K., "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, Vol. 2, pp. 559–572. 1901.
15. J. D. Hales, K. A. Gamble, B. W. Spencer, S. R. Novascone, G. Pastore, W. Liu, D. S. Stafford, R. L. Williamson, D. M. Perez, R. J. Gardner, *BISON Users Manual, BISON Release 1.2*, INL/MIS-13-30314, September 2015
16. J. D. Hales, R. L. Williamson, S. R. Novascone, G. Pastore, B. W. Spencer, D. S. Stafford, K. A. Gamble, D. M. Perez, W. Liu, *BISON Theory Manual, The Equations Behind Nuclear Fuel Analysis, BISON Release 1.2* Technical Report, INL/EXT-13-29930, September 2015
17. S. Djurle, *The Super-Ramp Project,* Report STUDSVIK-STSR-32, Studsvik AB Atomenergi Sweden, 1984.

# Appendix A

# RAVEN User Manual Updates

These are sections that have been updated or created as a result of RAVEN data mining activities (in bold the specific addition. The part specifically referring to the time dependent newly introduced part is underlined.

### 17.5.1 BasicStatistics

The **BasicStatistics** post-processor is the container of the algorithms to compute many important statistical quantities. It is important to note that this post-processor can accept as input either a ***PointSet*** or a ***HistorySet*** data object, depending on the type of statistics a user wants to compute:

- ***PointSet***: Static Statistics;
- ***HistorySet*****:** Dynamic Statistics. Depending on a "pivot parameter" (e.g. time) the post-processor is going to compute the statistics for each value of it (e.g. for each time step). In case an HistorySet is provided as Input, the Histories needs to be synchronized (use *Interfaced* post-processor of type HistorySetSync).

In order to use the *BasicStatistics post-processor*, the user needs to set the `subType` of a `<PostProcessor>` node:

```
<PostProcessor subType = 'BasicStatistics' />
```

Several sub-nodes are available:

• `<what>`, ***comma separated string, required field***, List of quantities to be computed. Currently the quantities available are:

- **covariance**: covariance matrix
- **NormalizedSensitivity**: matrix of normalized sensitivity coefficients. **Note:** It is the matrix of normalized **VarianceDependentSensitivity**
- **VarianceDependentSensitivity**: matrix of sensitivity coefficients dependent on the variance of the variables
- **sensitivity**: matrix of sensitivity coefficients, computed via linear regression method.
- **pearson**: matrix of correlation coefficients
- **expectedValue**: expected value or mean
- **sigma**: standard deviation
- **variationCoefficient**: coefficient of variation, i.e. **sigma**/**expectedValue**. Note: If the **expectedValue** is zero, the **variationCoefficient** will be INF.
- **variance**: variance
- **skewness**: skewness
- **kurtosis**: excess kurtosis (also known as Fisher's kurtosis)
- **median**: median

- **percentile**: the percentile. If this quantity is inputted as *percentile* the 5% and 95% percentile(s) are going to be computed. Otherwise the user can specify this quantity as *percentile* X%, where *X* represents the requested percentile (an integer value between 1 and 100)
- **samples**: the number of samples in the data set used to determine the statistics.
- **minimum**: The minimum value of the samples.
- **maximum**: The maximum value of the samples.

**Note:** If the weights are present in the system then weighted quantities are calculated automatically. In addition, if a matrix quantity is requested (e.g. Covariance matrix, etc.), only the weights in the output space are going to be used for both input and output space (the computation of the joint probability between input and output spaces is not implemented yet).

**Note:** Certain ROMs provide their own statistical information (e.g., those using the sparse grid collocation sampler such as: **'GaussPolynomialRom'** and **'HDMRRom'**), which can be obtained by printing the ROM to file (xml). For these ROMs, computing the basic statistics on data generated from one of these sampler/ROM combinations may not provide the information that the user expects.

If all the quantities need to be computed, the user can input in the body of **<what>** to the string `all`.

- **<biased>**, ***string (boolean), optional field***, if *True* biased quantities are going to be calculated, if *False* unbiased.
  *Default: False*
- **<parameters>**, ***comma separated string, required field***, lists the parameters on which the previous operations need to be applied (e.g., massFlow, Temperature)
- **<methodsToRun>**, ***comma separated string, optional field***, specifies the method names of an external Function that need to be run before computing any of the predefined quantities. If this XML node is specified, the **<Function>**, node must be present.
  *Default: None*
- **Assembler Objects** This object is required in case the **<methodsToRun>** node is specified. The object must be listed with a rigorous syntax that, except for the xml node tag, is common among all the objects. Each of these nodes must contain 2 attributes that are used to map those within the simulation framework:
  - `class`, ***required string attribute***, it is the main "class" the listed object is from
  - `type`, ***required string attribute***, it is the object identifier or sub-type

The **BasicStatistics** post-processor approach optionally accepts the following object type:

- **<Function>**, ***string, required field***, The body of this xml block needs to contain the name of an External Function defined within the **<Functions>** main block (see section 18). This object needs to contain the methods listed in the node **<methodsToRun>**.
- **<pivotParameter>**, ***string, optional field***, name of the parameter that needs to be used for the computation of the Dynamic BasicStatistics (e.g. time). This node needs to be inputted just in case a **HistorySet** is used as the input. It represents the reference monotonic variable based on which the statistics is going to be computed (e.g. time-dependent statistical moments).
- *Default: None*

**Example (Static Statistics):**

```
<Simulation>
  ...
  <Models>
    ...
    <PostProcessor name='aUserDefinedName' subType='BasicStatistics'
       verbosity='debug'>
      <!-- Here you can specify what type of figure of merit you need to
           compute: expectedValue, sigma, variance, kurtosis, pearson,
           covariance, etc. -->
      <what>expectedValue</what>
      <parameters>x01,x02</parameters>
      <methodsToRun>failureProbability</methodsToRun>
    </PostProcessor>
    ...
  </Models>
  ...
</Simulation>
```

**Example (Dynamic Statistics):**

```
<Simulation>
  ...
  <Models>
    ...
    <PostProcessor name='aUserDefinedNameForDynamicPP'
       subType='BasicStatistics' verbosity='debug'>
      <!-- Here you can specify what type of figure of merit you need to
           compute: expectedValue, sigma, variance, kurtosis, pearson,
           covariance, etc. -->
      <what>expectedValue,covariance,variance</what>
      <parameters>x01,x02</parameters>
    <methodsToRun>failureProbability</methodsToRun>
    <pivotParameter>time</pivotParameter>
    </PostProcessor>
    ...
  </Models>
  ...
</Simulation>
```

#### 17.5.9.1 SciKitLearn

**'SciKitLearn'** is based on algorithms in the SciKit-Learn library, and it performs data mining over **PointSet** and **HistorySet** objects. Note that for **HistorySet** objects **'SciKitLearn'** performs the task given in **<SKLType>** (see below) for each time step, and so only synchronized **HistorySet** objects can be used as input to this model. For an unsynchronized **HistorySet**, use the **'HistorySetSync'** method in the **'Interfaced'** post-processor to synchronize the input data before using a **'SciKitLearn'** post-processor. The rest of this subsection and the following subsection are dedicated to the **'SciKitLearn'** library.

The temporal variable for a **HistorySet** **'SciKitLearn'** is specified in the **<pivotParameter>** node:

- **<pivotParameter>**, ***string, optional parameter***, specifies the pivot variable (e.g., time, etc) in the input HistorySet.

  *Default: None*.

The algorithm for the dataMining is chosen by the subnode **<SKLType>** under the parent node **<KDD>**. The format is same as in 17.3.6. However, for the completeness sake, it is repeated here.

The data that are used in the training of the DataMining postprocessor are supplied with the subnode **<Features>** in the parent node **<KDD>**.

- **<SKLType>**, ***vertical bar (|) separated string, required field***, contains a string that represents the data mining algorithm to be used. As mentioned, its format is: **<SKLType>**`mainSKLclass|algorithm`**</SKLType>** where the first word (before the "|" symbol) represents the main class of algorithms, and the second word (after the "|" symbol) represents the specific algorithm.
- **<Features>**, ***string,requiredfield***, defines the data to be used for training the data mining algorithm. It can be:
  - The name of the variable in the defined dataObject entity
  - The location (i.e. input or output). In this case the data mining is applied to all the variables in the defined space.

The **<KDD>** node can have either optional or required subnodes depending on the dataMining algorithm used. The possible subnodes will be described separately for each algorithm below. The time dependent clustering data mining algorithms have a **<reOrderStep>** option that will try and keep the same labels on the clusters. A higher number implies a longer history that the clustering algorithm will look through to maintain the same labeling between time steps. All the available algorithms are described in the following sections.

#### 17.5.10.3 Method: HistorySetSync

This Post-Processor performs the conversion from a **HistorySet** object to another **HistorySet** object. The conversion is made so that all histories are synchronized in time (or some other user-defined pivot parameter). It can be used to allow the histories to be sampled at the same time instant.

There are two possible synchronization methods, specified through the **<syncMethod>** node. If the **<syncMethod>** is **'grid'**, a **<numberOfSamples>** node is specified, which yields an equally spaced grid of time points. The output values for these points will be linearly derived using nearest sampled time points, and the new **HistorySet** object will contain only the new grid points.

The other methods are used by specifying **<syncMethod>** as **'all'**, **'min'**, or **'max'**. For **'all'**, the postprocessor will iterate through the existing histories, collect all the time points used in any of them, and use these as the new grid on which to establish histories, retaining all the exact original values and interpolating linearly where necessary. In the event of **'min'** or **'max'**, the postprocessor will find the smallest or largest time history, respectively, and use those time values as nodes to interpolate between. In the **<PostProcessor>** input block, the following XML sub-nodes are required, independent of the **subType** specified:

- **<timeID>**, ***string, required field***, ID of the temporal variable
- **<extension>**, ***string, required field***, type of extension when the sync process goes outside the boundaries of the history (zeroed or extended)

**<syncMethod>**, ***string, required field***, synchronization strategy to employ (see description above). Options are **'grid'**, **'all'**, **'min'**, **'max'**.

**<numberOfsamples>**, ***integer, optional field***, required if **<syncMethod>** is **'grid'**, number of new time samples.

# Appendix B

# Regression Test Additions

## Basic Statistics Regression Test Additions

1. PostProcessors/BasicStatistics.testTimeDependentBasicStatistics
2. PostProcessors/BasicStatistics.testTimeDependentBasicStatisticsAsynchronousHistories

## Clustering Regression Test Additions

1. PostProcessors/TemporalDataMiningPostProcessor/Clustering.AffinityPropogation
2. PostProcessors/TemporalDataMiningPostProcessor/Clustering.DBSCAN
3. PostProcessors/TemporalDataMiningPostProcessor/Clustering.DirichletProcessGMM
4. PostProcessors/TemporalDataMiningPostProcessor/Clustering.GaussianMixture
5. PostProcessors/TemporalDataMiningPostProcessor/Clustering.KMeans
6. PostProcessors/TemporalDataMiningPostProcessor/Clustering.MeanShift
7. PostProcessors/TemporalDataMiningPostProcessor/Clustering.MiniBatchKMeans
8. PostProcessors/TemporalDataMiningPostProcessor/Clustering.SpectralClustering
9. PostProcessors/TemporalDataMiningPostProcessor/Clustering.VariationalGMM

## Dimensionality Reduction Regression Test Additions

1. PostProcessors/TemporalDataMiningPostProcessor/DimensionalityReduction.ExactPCA
2. PostProcessors/TemporalDataMiningPostProcessor/DimensionalityReduction.FastICA
3. PostProcessors/TemporalDataMiningPostProcessor/DimensionalityReduction.KernelPCA
4. PostProcessors/TemporalDataMiningPostProcessor/DimensionalityReduction.LocallyLinearEmbedding
5. PostProcessors/TemporalDataMiningPostProcessor/DimensionalityReduction.MiniBatchSparsePCA
6. PostProcessors/TemporalDataMiningPostProcessor/DimensionalityReduction.MultiDimensionalScaling
7. PostProcessors/TemporalDataMiningPostProcessor/DimensionalityReduction.RandomizedPCA
8. PostProcessors/TemporalDataMiningPostProcessor/DimensionalityReduction.SparsePCA
9. PostProcessors/TemporalDataMiningPostProcessor/DimensionalityReduction.TruncatedSVD