

Reactor Analysis and Virtual Control Environment (RAVEN) FY12 Report

Cristian Rabiti
Andrea Alfonsi
Joshua Cogliati
Diego Mandelli
Robert Kinoshita

September 2012



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

Reactor Analysis and Virtual Control Environment (RAVEN) FY12 Report

**Cristian Rabiti
Andrea Alfonsi
Joshua Cogliati
Diego Mandelli
Robert Kinoshita**

September 2012

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

<http://www.inl.gov>

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

EXECUTIVE SUMMARY

RAVEN is a complex software tool that has functionality spanning from being the RELAP-7 user interface, to using RELAP-7 to perform Risk Informed Safety Characterization (RISMC), and to controlling RELAP-7 calculation execution.

The goal of this document is to:

1. Highlight the functional requirements of the different tasks of RAVEN
2. Identify shared functions that could be aggregated into modules to obtain minimal software redundancy and maximize software utilization.

RAVEN is a software framework that will allow the user to take advantage of the following functionalities:

- Derive and implement the control logic required to:
 - Simulate the plant control system
 - Simulate the operator (procedure guided) actions
 - Perform Monte Carlo sampling of random distributed events
 - Perform event tree based analysis
- Provide a GUI to:
 - Input a plant description to RELAP-7 (component, control variable, control parameters)
 - Concurrent monitoring of control parameters
 - Concurrent alteration of control parameters
- Provide Post Processing data mining capability based on
 - Dimensionality reduction
 - Cardinality reduction

This document will show how an appropriate mathematical formulation of the control logic and probabilistic analysis leads to having most of the software infrastructure leveraged between the two main tasks.

Further, this document will go through the developments accomplished this year, including simulation results, and priorities for the next year's development.

0 Introduction

The document is divided in five main sections:

1. Control Logic

The analysis of the control logic classical equation set is carried over in a very general formalism and then the software infrastructure is derived

2. GUI

The requirements of the GUI are highlighted and integrated into the software design

3. Probabilistic safety analysis

A general mathematical framework for modeling dynamic stochastic systems is introduced. Then, the software implementation is derived so to maximize leveraging with the control logic framework

4. Results

An overview of results obtained so far is presented: GUI capabilities, control logic application and probabilistic analysis

5. Future developments

For each of the above-defined tasks of RAVEN, the main future tasks and challenges are outlined

6. DEMO set up

The initial preparation of a DEMO based on a PWR for PRA analysis is described

1 Control Logic

1.1 Plant and Control System Model

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

$$\frac{\partial \bar{\theta}(t)}{\partial t} = \bar{\mathcal{H}}(\bar{\theta}(t), t)$$

Eq. 1

In the above equation we have assumed the time differentiability of the phase space. This is generally not required and it is used here for compactness of the notation.

Now an arbitrary decomposition of the phase space is performed:

$$\bar{\theta} = \begin{pmatrix} \bar{x} \\ \bar{v} \end{pmatrix}$$

Eq. 2

This decomposition is made in such a way that \bar{x} represents the set of unknowns solved by RELAP-7, while \bar{v} represents the variables directly controlled by the control system.

Eq. 1 is now cast in a system of equations:

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

Eq. 3

As a consequence of this splitting, \bar{x} contains only the state variables of the phase space that are continuous while \bar{v} contains only the discrete state variables that are usually handled by the control system. Next we notice that the function $V(\bar{x}, \bar{v}, t)$ representing the control system, does not depend on the knowledge of the complete status of the system but on a restricted subset that we call control variables \bar{C} :

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

Eq. 4

where \bar{C} is a vector having lesser dimensionality than \bar{x} and therefore is more convenient to work with. In this document the following naming convention will be used:

- \bar{C} : Monitored variables
- \bar{v} : Controlled variable

Note that even if it seems more appropriate, the standard naming system of signals (monitored) and status (controlled) is not used. The reason for this choice is that, the chosen naming system better mirrors the computational pattern between RAVEN and RELAP 7. Moreover, the definition of signals is tight to the definition of the control logic for each component and therefore relative rather than absolute in the overall system analysis. In fact we could have signal for a component that is the status of another component creating a definition that would be not unique. Another reason is that the standard naming will loose every meaning once used also for uncertainty analysis.

1.2 Operator Splitting Approach for the Simulation of the Control System

The system of equations shown in Eq. 4 is, generally speaking, fully coupled and in the past it has always been solved using an operator splitting approach. The reasons for this choice are several:

- Control systems react with an intrinsic delay
- The reaction of the control system might move the system between two different discrete state and therefore numerical errors will be always of first order unless the discontinuity is treated explicitly

At least in its initial implementation, RAVEN will follow this common practice. Therefore Eq. 4 becomes:

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

Eq. 5

1.3 Operator Response

It will not be RAVEN's task to define mathematical models of plant operator behavior, but at the same time once those models (for the moment only deterministic) are built it would be easy to implement them in a similar fashion to the implementation of the plant control system itself.

1.4 Software Implementation of the Control System

RELAP-7 is the solver for the plant system except for the control system. From the mathematical formulation presented so far, RELAP-7 will solve $\frac{\partial \bar{x}}{\partial t} = \bar{F}(\bar{x}, \bar{v}_{t_{i-1}}, t)$. RELAP-7 will be based on the middleware software MOOSE, that, in addition to providing the algorithms for the solution of the differential equation, will also provide all the manipulation tools for the C++ classes containing the solution vector. More specifically, the plant will be represented by a set of components and each component type will correspond to a C++ class. At each time step RELAP-7/MOOSE will update the information within the classes with the current solution \bar{x} , then RAVEN will ask MOOSE to perform the needed manipulation to construct the monitored quantities \bar{C} . Once \bar{C} is constructed, the information is reduced to a vector of numbers understandable to the control system. The equation $\frac{\partial \bar{v}}{\partial t} = \bar{V}(\bar{C}, \bar{v}_{t_{i-1}}, t)$ is solved and the set of control parameters for the next time step (\bar{v}_{t_i}) is obtained. Up to now no situations where the complexity of $\frac{\partial \bar{v}}{\partial t} = \bar{V}(\bar{C}, \bar{v}_{t_{i-1}}, t)$ required a numerical solution therefore for the moment RAVEN remains numerical integration free. Note that once the information is transferred to \bar{C} , the way the plant solution (\bar{x}) is computed or stored is irrelevant. The last statement highlights the capability of RAVEN to represent an easily generalizable tool. To be more specific, in reality MOOSE is made aware of the need to compute at the end of each time step the \bar{C} . As a consequence this is computed before returning the calculation to RAVEN. Therefore, the scheme in Figure 1 is more accurate in terms of software implementation.

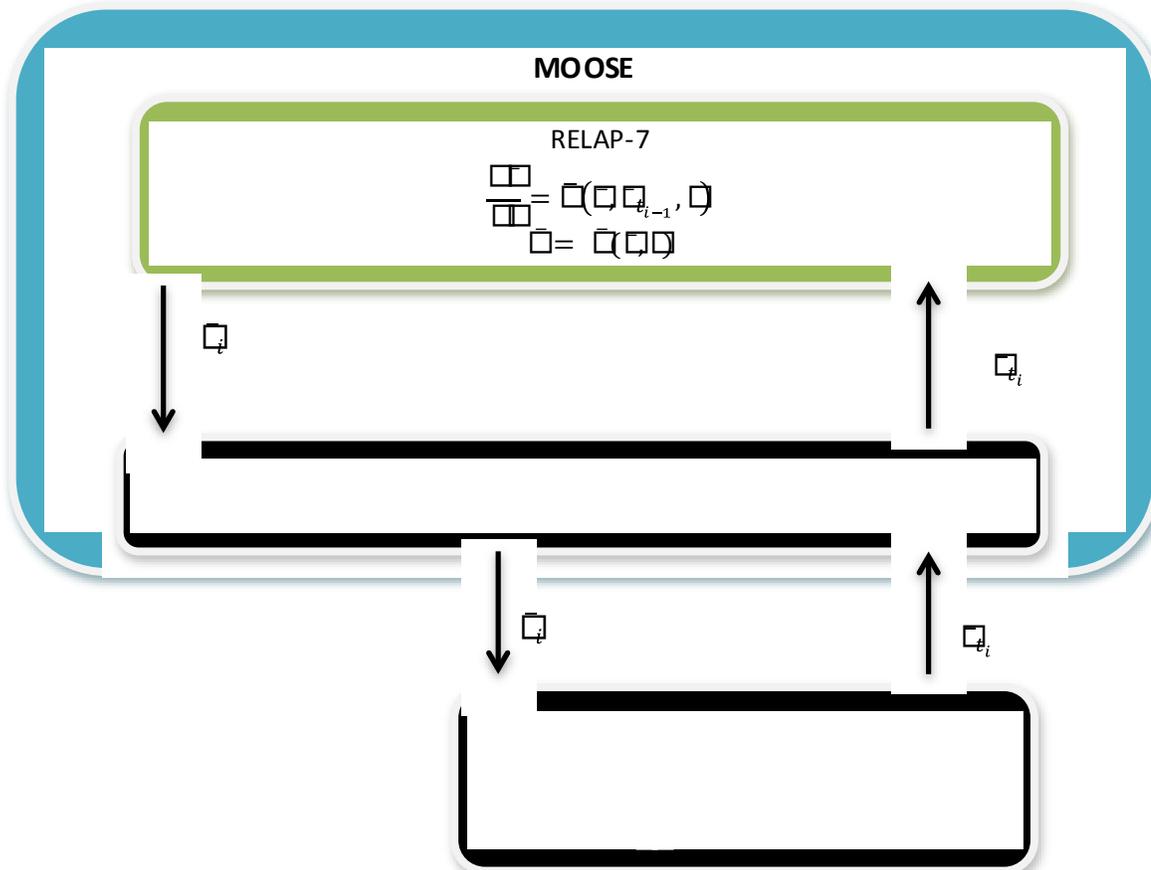


Figure 1: Control system software layout

1.5 Definition of the \bar{C} Space

The contraction of the information from the \bar{x} space to the \bar{C} space is a crucial step that defines the functional requirements of both modules: Control Logic and RELAP-7 Interface. Since \bar{C} represents an arbitrary middle step, a set of rules are defined that will make this choice unique. At the same, it would make sense to use this degree of freedom to minimize software redundancy and clear methods of ownership (e.g where algorithms are implemented and maintained). After accounting for the above considerations, \bar{C} will be chosen such that:

- The solution of $\left. \frac{\partial \bar{v}}{\partial t} \right| = \bar{V}(\bar{C}, \bar{v}_{t_{i-1}}, t)$ could be carried along without any knowledge of the solution algorithm of $\left. \frac{\partial \bar{x}}{\partial t} \right| = \bar{F}(\bar{x}, \bar{v}_{t_{i-1}}, t)$.
- *This sets the minimum contraction of the information from \bar{x} to \bar{C}*
- All the actions represented by $\bar{C} = \bar{G}(\bar{x}, t)$ require knowledge of the solution algorithm of $\left. \frac{\partial \bar{x}}{\partial t} \right| = \bar{F}(\bar{x}, \bar{v}_{t_{i-1}}, t)$, i.e. the derivation of $\bar{C} = \bar{G}(\bar{x}, t)$ is performed by subsequent calls to the MOOSE post processing capabilities.
- *This set the maximum contraction of the information from \bar{x} to \bar{C}*

The intersection of the two subspaces above defined seems to create a unique minimal set (at least in all the problems sampled so far).

At this point an example might be helpful:

- Controlled parameter: bypass valve status (open/closed, 1/0)
- Control logic: differential pressures between two specific points along pipes i and j set the opening of the bypass

Let:

- $p_i = p_i(l)$, $0 \leq l \leq L_i$, pressure along pipe i
- $p_j = p_j(l)$, $0 \leq l \leq L_j$, pressure along pipe j
- $s = \begin{cases} 1 \\ 0 \end{cases}$, control variable of the bypass
- l_i , coordinate of the control point of the pressure along pipe i
- l_j , coordinate of the control point of the pressure along pipe j
- ΔP_o , opening delta pressure
- ΔP_c , closing pressure drop

The control logic becomes:

$$if\ s = \begin{cases} 0, & then \begin{cases} \int_0^{L_i} dl\delta(l_i - l)p_i(l) - \int_0^{L_j} dl\delta(l_j - l)p_j(l) \geq \Delta P_o & then\ s = 1 \\ \int_0^{L_i} dl\delta(l_i - l)p_i(l) - \int_0^{L_j} dl\delta(l_j - l)p_j(l) < \Delta P_o & then\ s = 0 \end{cases} \\ 1, & then \begin{cases} \int_0^{L_i} dl\delta(l_i - l)p_i(l) - \int_0^{L_j} dl\delta(l_j - l)p_j(l) \leq \Delta P_c & then\ s = 0 \\ \int_0^{L_i} dl\delta(l_i - l)p_i(l) - \int_0^{L_j} dl\delta(l_j - l)p_j(l) < \Delta P_c & then\ s = 1 \end{cases} \end{cases}$$

Following the above specified role for the creation of \bar{C} we introduce:

$$C_m = \int_0^{L_i} dl\delta(l_i - l)p_i(l) \text{ control variable } m$$

$$C_n = \int_0^{L_j} dl\delta(l_j - l)p_j(l) \text{ control variable } n$$

That in a coherent notation could be translated in the following expression for \bar{G} :

$$\bar{C} = \begin{pmatrix} C_m \\ C_n \end{pmatrix} = \bar{G}(\bar{x}, t) = \begin{pmatrix} \int_0^{L_i} dl\delta(l_i - l)p_i(l) \\ \int_0^{L_j} dl\delta(l_j - l)p_j(l) \end{pmatrix}$$

And also the formulation of the control logic would become:

$$if\ s = \begin{cases} 0, & \text{then } \begin{cases} C_m - C_n \geq \Delta P_o & \text{then } s = 1 \\ C_m - C_n < \Delta P_o & \text{then } s = 0 \end{cases} \\ 1, & \text{then } \begin{cases} C_m - C_n \leq \Delta P_c & \text{then } s = 0 \\ C_m - C_n < \Delta P_c & \text{then } s = 1 \end{cases} \end{cases}$$

1.6 The Auxiliary Plant and Component Status Variable

So far it is assumed that all information needed is contained in \bar{x} and \bar{v} . Even if, as previously shown, this information is sufficient for the calculation of the system status in every point in time it is not a very practical way to implement the control system.

To focus the idea it might be helpful to consider the following example.

- Pump A has two different behaviors (head value) depending if in a pre scram or post scram situation or pump fault.
- In this case to decide what would be its head at a certain point of the simulation the whole history of the plant and of the pump would be needed to decide if either a fault or scream condition has been encountered.
- In order to avoid this inefficiency it is possible to introduce a set of auxiliary variables \bar{a} that do not add information but simplify the implementation of the control system.

Later in the report there will be a more deep discussion on what is the mathematical meaning of this variables but essentially those are the variables that in statistical analysis are artificially added to the phase space to force a Markovian behavior into the system, (i.e. only the previous time step information are needed to determine the future status of the system).

For convenience it is also useful to introduce two types (what is called here convenience is in a more correct mathematical wording for minimization of the control logic set):

- Global status auxiliary control variables, e.g., scram, time from scram initiated, hot shut-down, time from hot shut-down, cold shut-down, etc.
- Component status auxiliary variables, e.g., correct functioning status, time from fault, etc.

It could be helpful to make an example for the control law of a pump:

- Pump gets requested only after scram
- Pump stops properly working above a certain temperature cumulative load
- If the pump is active, the flow rate is a function of the temperature in the core
- If pump is not working, the flow rate decays exponentially with a certain lambda from the failure time

Following the above logic to perform the control of the pump we need:

- Control variables (\bar{C} space): cumulative temperature load
- Global auxiliary: scram status (1/0), scram time
- Component auxiliary: fault status (1/0), flow rate when fault occurred, time when fault occurred
- Component control parameter: flow rate

In conclusion, when building the input for the control system we need to:

- Define a set of global status auxiliary control parameters, initial values and control logic
- Define the component auxiliary control parameters initial values and control logic

It is important to notice that the sequence in which the global and local auxiliary, and the component control parameters are computed might alter the outcome of the simulation. The addition of a priority property to the control logic is therefore suggested as a future development. This choice might avoid closed loops and linearize in time the control logic.

To make an example we can just refer to the previous one where the control variable of the bypass ($s = \begin{cases} 1 \\ 0 \end{cases}$) now would be a component status auxiliary variable. We can also reformulate system Eq. 5 to account for this new set of auxiliary information.

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

Eq. 6

1.7 Example: Implementation of a Proportional Integrative and Differential (PID) Controller

A PID controller is one of the most general formulations for a controller therefore is considered a good example on which perform a test of the framework designed so far.

The mathematical representation is:

$$\frac{dv}{dt} = \alpha u(t) + \beta \frac{dy(t)}{dt} + \gamma \int_0^t dt' z(t')$$

Eq. 7

To highlight how the software framework is supposed to handle this type of controller it is helpful to assume:

$$u(t) = y(t) = z(t)$$

Eq. 8

And

$$u(t) = \int_{V_i} d\bar{r} P(\bar{r}, t)$$

Eq. 9

So that $u(t)$ represents the average pressure in component i . Eq 7 could be rewritten as:

$$\frac{dv}{dt} = \alpha \int_{V_i} d\bar{r} P(\bar{r}, t) + \beta \frac{d}{dt} \int_{V_i} d\bar{r} P(\bar{r}, t) + \gamma \int_0^t dt' \int_{V_i} d\bar{r} P(\bar{r}, t')$$

Eq. 10

Or

$$\frac{dv}{dt} = \alpha \int_{V_i} d\bar{r} P(\bar{r}, t) + \beta \int_{V_i} d\bar{r} \frac{d}{dt} P(\bar{r}, t) + \gamma \int_{V_i} d\bar{r} \int_0^t dt' P(\bar{r}, t')$$

Eq. 11

It is clear from last formulation that, given the rule described in the previous paragraph for the definition of the \bar{C} space, all the terms appearing in the above equation need to be treated as a separate C type variables. This is due to the fact that RAVEN is blind with respect the numerical scheme in time used by MOOSE. Once we introduce the operator splitting approach, the right side of Eq. 11 is a constant therefore the final status will be simply:

$$v_t = (\alpha C_{1,t_{i-1}} + \beta C_{2,t_{i-1}} + \gamma C_{3,t_{i-1}})t + v_{t-1}$$

Eq. 12

The task repartitioning between MOOSE and RAVEN is depicted in the figure below:

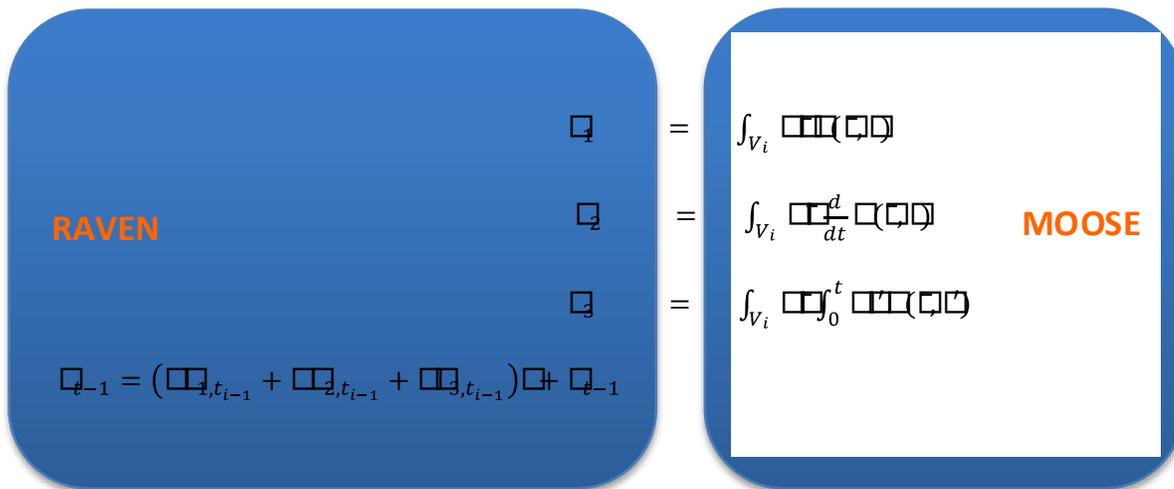


Figure 3: Task repartitioning for the PID implementation

1.7.1.1 Kernel (Operators) Needed to Create the Necessary Control Variables

In order to do so the following list of operators needs to be supported by the MOOSE framework (priority ordered):

$$\frac{1}{\bigcup_{i=1}^n V_i} \int_{\bigcup_{i=1}^n V_i} d\bar{r}$$

$$\max\{\cdot | \forall \bar{r} \in \bigcup_{i=1}^n V_i\}$$

$$\min\{\cdot | \forall \bar{r} \in \bigcup_{i=1}^n V_i\}$$

$$\int_{\bigcup_{i=1}^n V_i} \delta(\bar{r} - \bar{r}_0) d\bar{r}$$

The average operator should be provided both for each single component (n=1) and for a combination of them (this is a lower priority) since this type of averages need to be volume weighted (already implemented)

The min/max operator for the moment could be limited to the maximum on the nodal point while in the future it might be required to compute a more accurate value (Gaussian node, or arbitrary point search). As before the extension to multiple components could be a feature added later

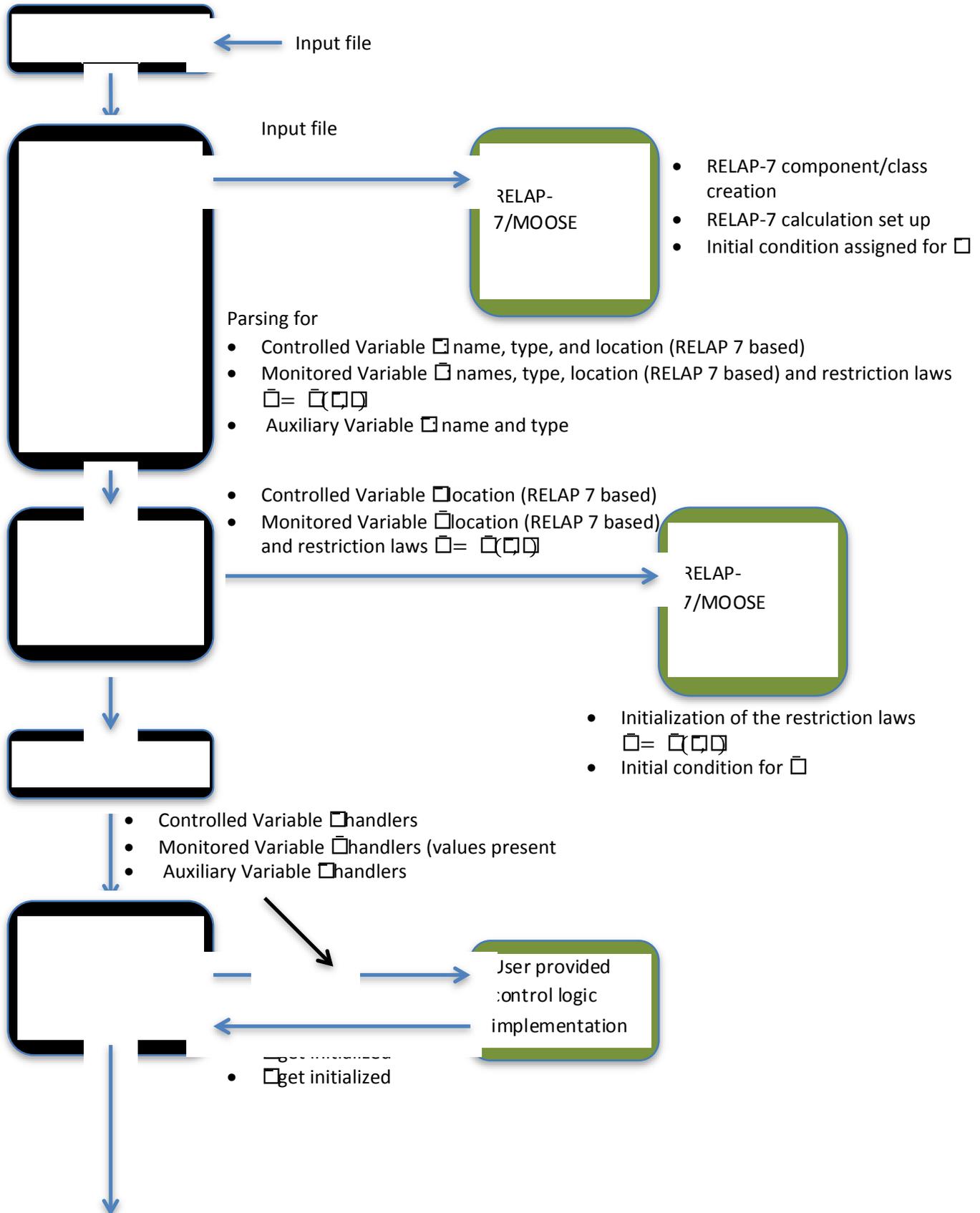
The point wise operator is especially needed when it is relevant to match the physical location of transponders with the computed controlled quantity

$$\frac{d}{dt} \begin{cases} \frac{1}{\cup_{i=1}^n V_i} \int_{\cup_{i=1}^n V_i} d\bar{r} \\ \max\{ \cdot |\forall \bar{r} \in \cup_{i=1}^n V_i \} \\ \min\{ \cdot |\forall \bar{r} \in \cup_{i=1}^n V_i \} \\ \int_{\cup_{i=1}^n V_i} \delta(\bar{r} - \bar{r}_0) d\bar{r} \end{cases} \quad \text{Time derivative operator}$$

$$\int_{t_0}^{t'} dt' \begin{cases} \frac{1}{\cup_{i=1}^n V_i} \int_{\cup_{i=1}^n V_i} d\bar{r} \\ \max\{ \cdot |\forall \bar{r} \in \cup_{i=1}^n V_i \} \\ \min\{ \cdot |\forall \bar{r} \in \cup_{i=1}^n V_i \} \\ \int_{\cup_{i=1}^n V_i} \delta(\bar{r} - \bar{r}_0) d\bar{r} \end{cases} \quad \text{Time integral operator}$$

1.8 Control Logic Software Layout

The following scheme represents the current implementation that has been built to cover also all foreseen future needs.



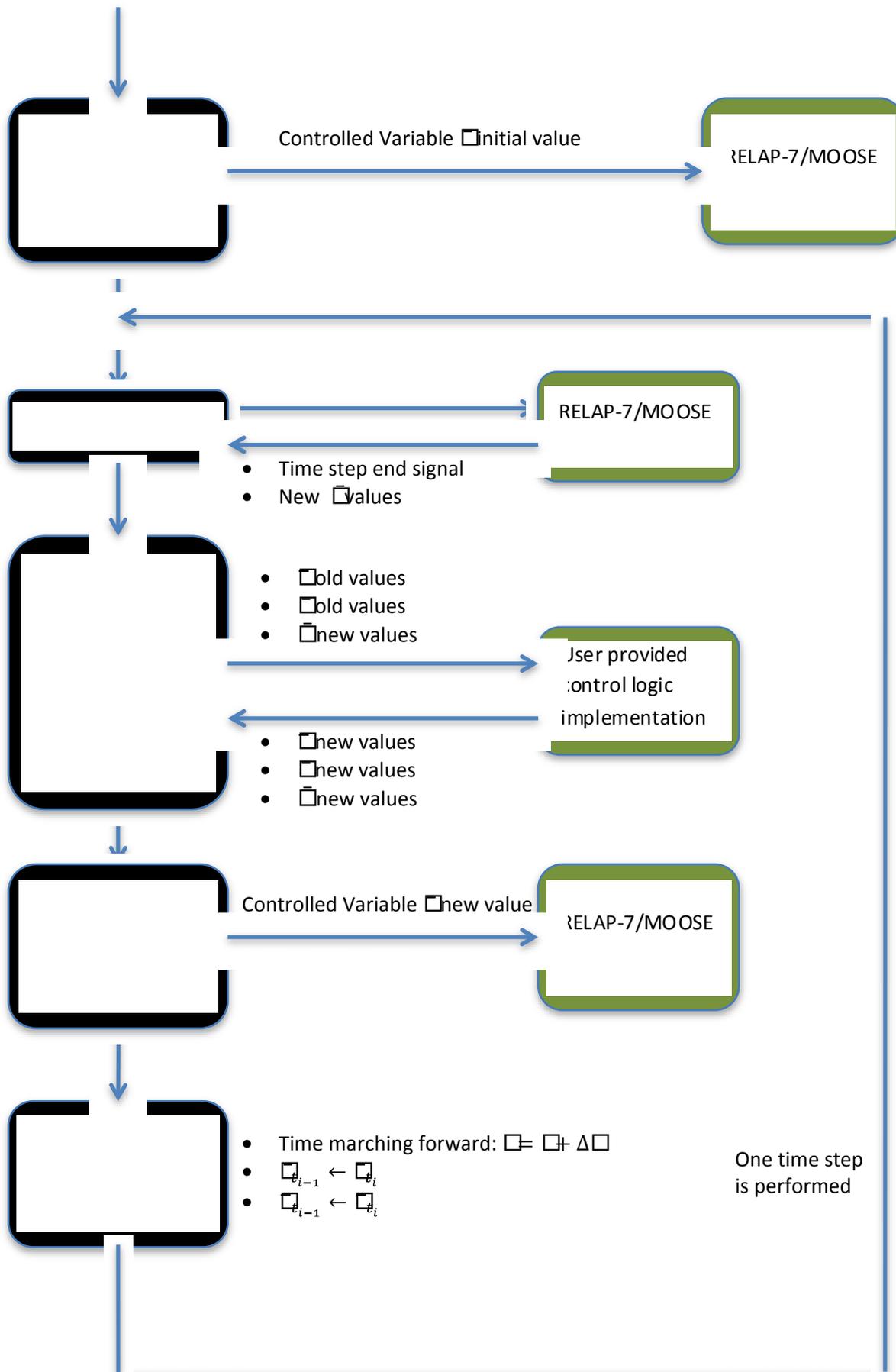


Figure 2 Calculation Flow

1.9 Control Logic Final Remarks

1.9.1 Control Logic User interface

So far the control logic is implemented by the user via Python scripting. The reason of this choice has been to try to preserve generality of the approach in the initial phases of the project so that later specialization would be possible and less expensive. At the moment the variables needed to define the control logic are accessible via an interface transparent to the user and once they are defined in the input file they are accessible in the control logic in the following form:

- Auxiliary.name
- Controlled.name
- Monitored.name

While it is believed that this is a rather convenient form to work with, it will be part of the GUI task to also automatize the construction of the control logic scripting to minimize user effort. Already available is the possibility to define via the input the creation of customizable distribution and cumulative distribution functions. Once the user provide names to those specialization they will also available in the python scripting in a similar form to the other variables (distribution.name(allowable list of arguments)).

1.9.2 Concerning Usage and Definition of Auxiliary Variable

While the addition of auxiliary variables is a requirement to recast the system in a Markovian form during our preliminary test it has been noticed that they are useful tools to simplify the implementation of the control logic from a user standpoint. This usage increases the dimension of the phase space above what is strictly needed to represent the system but so far it has not been considered an issue.

2 Graphical User Interface

The Graphical User Interface has been highlighted by several of our stakeholders as one of the key success factors of the whole project (also including RELAP-7 itself). The GUI will evolve along with the expansion of the code capabilities therefore at this stage is very important to maintain the project open to future expansion and modular. Feedback from the stakeholder suggest the below discussed capabilities of the GUI to be prioritized.

2.1 GUI Capability Requirements

2.1.1 Graphical input of plant components and overall overview of the plant

For each component the GUI provides a box to input:

- Full component description
- The list of the needed inputs required to fully describe a RELAP-7 component is available via an automatized process. Each new defined component has to inform the underling software (MOOSE) of the input need to fully describe itself. This information is retrievable by a special command in MOOSE that is called at the initialization stage of the GUI, so that the construction of new input interface for a new component is completely automatized.

Next requirement is the capability to set up the general calculation requirement (accuracy, solver type, ending time etc.) like:

- Algorithm choice
- Convergence criteria
- Ending time

Note that also this process benefit of the above description automatic process.

The above capabilities are exhaustive with respect the creation of a new RELAP-7/MOOSE input file and of course there should be the capability to open and modify also a pre-existing input file

As long as the development of the GUI proceed to easy the user task we envision:

- a 3D layout in real geometry
- a 2D projection (in a fake geometry avoiding component overlapping)
- the possibility to inquire each component eventually modify it
- Drag and drop component
- CAD file importing

The creation of a 2D projection of the plant, while is at the present the most common visualization layout for similar software, poses here a quite difficult challenge since the general framework under which RELAP 7 is developed will easier provide support for the 3D view. Unfortunately the 3D view might become unpractical for very complex geometry and therefore later in the project it might be required to provide also the 2D visualization capability.

2.1.1.1 Graphical input of the control system

After the building of the physical layout of the plant it is possible to move forward toward the creation of the information necessary to define the control system.

To properly built the control logic the following information will be necessary:

- List of the control variable \bar{C} . A possibility to import such a list from the RELAP-7/MOOSE file (if already present) should be provided
- List of the control variable \bar{v} . A possibility to import such a list from the RELAP-7/MOOSE file (if already present) should be provided
- List of the control variable \bar{a}
- Laws $\bar{C} \rightarrow \bar{a}$ and $\bar{C}, \bar{a} \rightarrow \bar{v}$. To be noticed that the logic at the beginning will be provide by a python scripting while later on it is foreseeable to be automatized via the GUI.

2.1.1.2 Online Monitoring

All the control variable, controlled parameters and auxiliary ones are available in an comma separated value file (CSV) and update while the calculation is running by the RAVEN (control) module. At the same time also the values of the variable computed by RELAP-7/MOOSE are stored in a separate exodus file. The online monitoring should be performed on both files. The GUI should provide the capability to the user to inspect all these variables. While the controlled, monitored and auxiliary variable are a 1D field therefore of easily visualization the solution field provided by RELAP-7 is still 1D but in three-dimensional layout (following plant layout). The visualization of this latter information will be therefore more complex and leveraging already developed software will be suggested. As part of the online monitoring of the simulation the user should also be able to interrupt the simulation and modify the control logic.

2.2 GUI Development environment

To develop a full GUI from scratch is a long task and most likely not needed. For this reason several possible option has been preventively evaluated. The development of the GUI itself will be performed using QtPy that is Python interface for a C++ based library (Qt) for GUI development, probably the most currently used environment for GUIs. The evident advantage of this choice is to have a state of the art environment and hopefully a higher chance to find people expert to speed the development.

Even adopting Qt the visualization of 3D dimensional field and giving the user post processing capabilities remain a large and difficult task. In order to optimize effort in this direction this development will take advantage of the native support for exodus file in the VTK libraries. This choice will ease the visualization task while providing the needed post processing capabilities remains still of difficult reach. One of the possibilities considered is to leverage the capability of PARAVIEW. PARAVIEW is a software developed by KitWare based on vtk libraries and implemented in Qt. This software has great specialization capabilities and also provides a very flexible and complete post processing capability. It will be part of the next fiscal year task to better define the user needs in this respect and identify the most effective strategy.

2.3 Software layout of the GUI-RAVEN interaction with the simulation flow

Since RAVEN RELAP 7 will run with or without the GUI interface and also the GUI could be used only for generating input files to be run later on, the coupling scheme should be designed to allow these different configuration. Another important point to be considered is that most likely there will be a regulatory requirement to preserve input files.

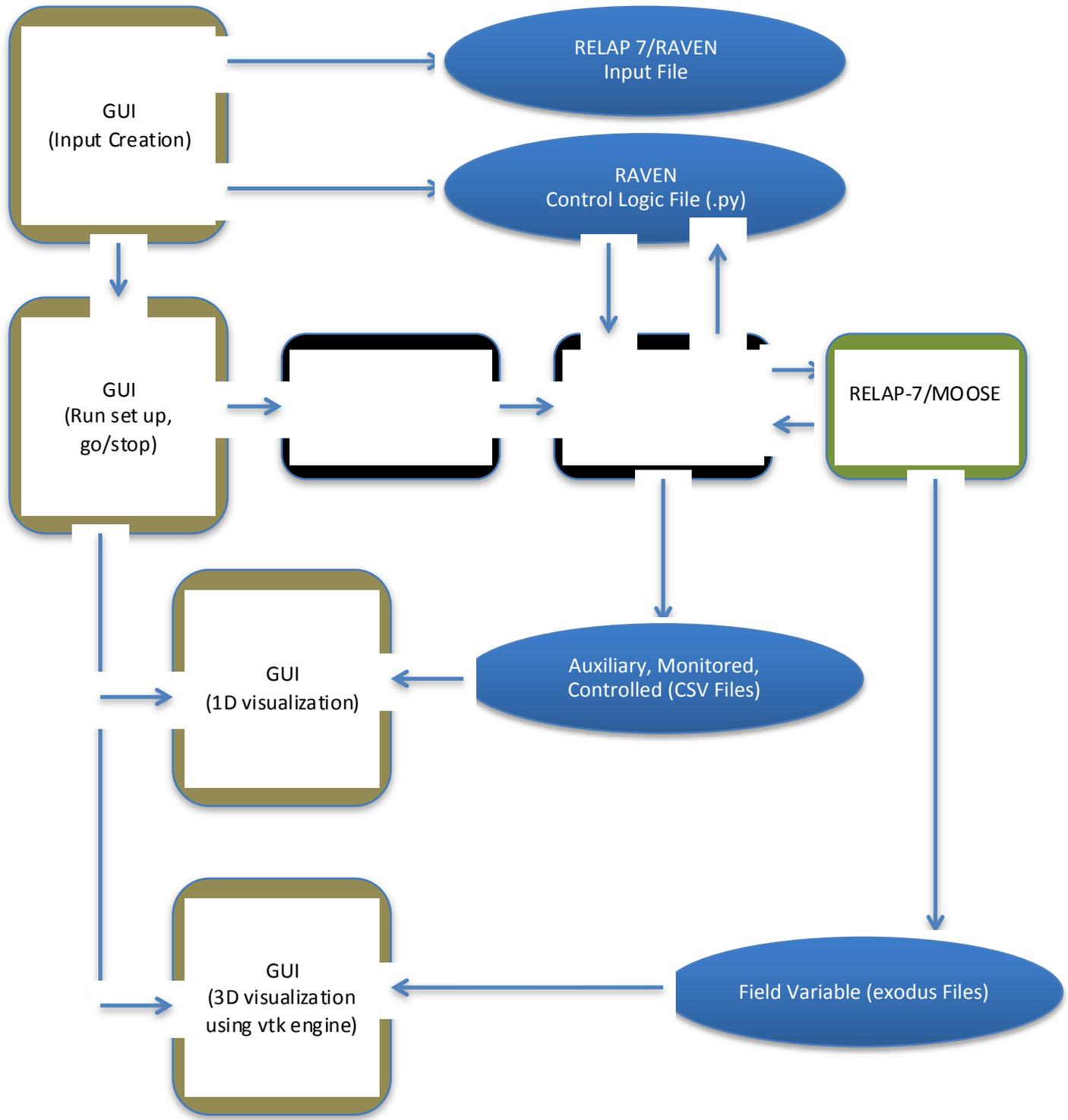


Figure 3: GUI and RAVEN driver interaction

3 RAVEN FOR MODELLING OF DYNAMIC STOCHASTIC SYSTEMS

3.1 Mathematical Framework

3.1.1 Derivation of the set of equations

The scope of this part of the document is to present the mathematical framework underling RAVEN development to model dynamic stochastic system. This framework also account for uncertainties. By dynamic stochastic systems we refer to systems whose dynamics contain random (i.e., not predictable a priori) elements. Random behaviors, although present in nature, it is often artificially introduced into models to account for incapability of producing simulations that accurate enough.

Although this differentiation, which pairs with the classical definition of epistemic (artificial) and aleatory (intrinsic) makes no difference in the way the evaluation on the uncertainties on the results of simulations. For the simulation of the behavior of those system, it will be more relevant the time dependence that those variable posses, this is anyhow a concept that will more deeply investigated later in this document.

Possible examples of random elements are:

- Random variability of parameters (e.g. uncertainty in physical parameters)
- Presence of noise (background noise due to intrinsically stochastic behaviors or lack of detail in the simulation)
- Uncertainty in the initial and boundary conditions
- Random failure of components
- Aging effect

While RAVEN's primary field of action is the reliability analysis of nuclear power plants when coupled with RELAP-7^a code, the concepts investigated in this section are of general validity for the analysis of dynamic stochastic systems. It might be helpful to first recall Eq. 1 where the initial condition is added:

$$\begin{cases} \frac{\partial \bar{\theta}(t)}{\partial t} = \bar{\mathcal{H}}(\bar{\theta}(t), t) \\ \bar{\theta}(t_0) = \bar{\theta}_0 \end{cases}$$

Eq. 13

Now each source of uncertainty or stochastic behavior will be considered and introduced one by one.

It is helpful also to split the space phase between continuous (e.g., temperature and pressure) and discontinuous variables (e.g., status of components including both operating and failure states), where the latter, as already discussed, are treated only by RAVEN.

- $\bar{\theta}^c \in \Phi \subseteq \mathbb{R}^C$ the set of continuous variables, and,
- $\bar{\theta}^d \in \Psi \subseteq \mathbb{N}^D$ the set of discrete variables (e.g., status of the system components).

Eq. 13 takes now the following form:

^a Even if RAVEN is designed to be directly coupled to RELAP-7, also other system simulators such as MELCOR [8], MAACS [5] and RELAP-5 [11] could be coupled to RAVEN.

$$\begin{cases} \frac{\partial \bar{\theta}^c(t)}{\partial t} = f(\bar{\theta}^c, \bar{\theta}^d, t) \\ \frac{\partial \bar{\theta}^d(t)}{\partial t} = g(\bar{\theta}^d, \bar{\theta}^c, t) \\ \bar{\theta}^c(t_0) = \bar{\theta}_0^c \\ \bar{\theta}^d(t_0) = \bar{\theta}_0^d \end{cases}$$

Eq. 14

For simplicity of notation we use the time derivative also for time discontinuous variables that mathematically is allow only after introducing very complex extension of the original time derivative operator.

The first stochastic behavior to be introduced is the uncertainty associated to the initial conditions and to the parameters defining the equations $f(\bar{\theta}^c, \bar{\theta}^d, t)$ and $g(\bar{\theta}^d, \bar{\theta}^c, t)$.

$$\begin{cases} \frac{\partial \bar{\theta}^c(t)}{\partial t} = f(\bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, t) \\ \frac{\partial \bar{\theta}^d(t)}{\partial t} = g(\bar{\theta}^d, \bar{\theta}^c, \bar{\alpha}_{Staz}, t) \\ \Pi(\bar{\theta}^c, t_0) \sim pdf(\bar{\theta}_0^c |, \sigma_c^2) \\ \Pi(\bar{\theta}^d, t_0) \sim pdf(\bar{\theta}_0^d |, \sigma_d^2) \\ \bar{\alpha}_{Staz}(t) = \bar{\alpha}_{Staz}(t_0) \sim pdf(\alpha_{Staz}^0 |, \sigma_{Staz}^2) \end{cases}$$

Eq. 15

Where the notation $pdf(m, \sigma^2)$ stand for the probability distribution function with mean value m and sigma σ , $\bar{\alpha}_{Staz}$ is the vector of parameters affected by uncertainty but not varying over time.

Up to now, we have considered uncertainties whose values do not change during the simulation. This set of uncertainties accounts for most of the common source of aleatory behaviors. Examples of this kind of uncertainties are:

- *uncertainty associated to heat conduction coefficient.* This value is known (but uncertain) has no physical reason to change during the simulation therefore follow within this category
- *Uncertainty on temperature for pipe failure.* This value is usually characterized by a probability distribution function but again one the value has been set (through random sampling) it will not change during the simulation. In fact it would be unphysical to not fail the pipe at 500K while, failing the same pipe at a lower temperature later on during the same simulation.

Note that up to now, by adding new variables, the dimensionality of the phase space has not increased and, moreover, the Markov property of the system is preserved.

The next aleatory component to be accounted for is the set of parameters continuously changing over time. To make an easy parallel it will be referred to this value as subject to a Brownian motion. While what commonly goes under the description of Brownian motion has not impact at the space and time scales characteristics of reactor plant simulation there are parameters that have or appear to have a similar behavior. The word appear is used to highlight that this type of behavior could be not natural but generated by lack of detail in the models. To fix ideas two examples might be helpful

- *Cumulative damage growth in materials.* While experimental data and models representing this phenomenon show large uncertainties there is also an intrinsic natural stochasticity driving the accumulation of the damage
- *Heat conductivity in the fuel gap during heating up of fuel.* During transient there are situation when the fuel is contact with the clad while other where there is the presence of a gap. While in nature this is locally a discontinuous transition it not usually possible to model in such a way, especially if vibration of the fuel lead to high frequency transitions. In such a case it would be helpful to introduce directly into the simulation a random noise characterizing the thermal conductivity in when these transition occur

The system Eq. 15 now should be rewritten in the following form:

$$\left\{ \begin{array}{l} \frac{\partial \bar{\theta}^c(t)}{\partial t} = f(\bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t) \\ \frac{\partial \bar{\theta}^d(t)}{\partial t} = g(\bar{\theta}^d, \bar{\theta}^c, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t) \\ \frac{\partial \bar{\alpha}_{Brown}}{\partial t} = b(\bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t)\Gamma(t) \\ \Pi(\bar{\theta}^c, t_0) \sim pdf(\bar{\theta}_0^c |, \sigma_c^2) \\ \Pi(\bar{\theta}^d, t_0) \sim pdf(\bar{\theta}_0^d |, \sigma_d^2) \\ \bar{\alpha}_{Staz}(t) = \bar{\alpha}_{Staz}(t_0) \sim pdf(\bar{\alpha}_{Staz}^0 |, \sigma_{Staz}^2) \\ \bar{\alpha}_{Brown}(t_0) \sim \bar{\alpha}_{Brown}^0 \Gamma(t_0) \end{array} \right.$$

Eq. 16

where $\Gamma(t)$ is 0-mean random noise and $\bar{\alpha}_{Brown}$ is the set of parameters subject to Brownian behavior.

Clearly the equation referring to the time change of the parameters subject to the *Brownian* motion should be interpreted in the Ito sense.

Last and probably most difficult step is the introduction of parameters that neither are constant during the simulation or continuously vary over time. Unfortunately this is still a quite common case. To fix idea lets' consider a valve that, provided set of operating conditions, opens or closes. If this set of conditions are reached n times during the simulation, the probability of the valve to correctly operate should be sampled n times. It is also foreseeable that the history of failure/success of the valve will impact future probability of failure/success. In this case the time evolution of such parameters (discontinuously stochastic changing parameters $\bar{\alpha}_{DS}$) is governed by the following equation.

$$\frac{\partial \bar{\alpha}_{DS}}{\partial t} = \bar{U}(\bar{\alpha}_{DS}, \bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t) \bar{V}(\bar{\alpha}_{DS}, \bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t) \bar{p} \left(\int_{t_0}^t dt \bar{\alpha}_{DS}, \bar{\alpha}_{DS}, \bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t \right)$$

Eq. 17

The above formula requires of course more in deep explanation.

- The function $\bar{\delta}(\bar{\alpha}_{DS}, \bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t)$ is the delta of Dirac of the instant on which the transition need to be evaluated (control logic signaling to the valve to open/close)
- The term $\bar{p} \left(\int_{t_0}^t dt \bar{\alpha}_{DS}, \bar{\alpha}_{DS}, \bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t \right)$ represents the transition probability between different states (in case of the valve open/close given the control logic signal). To be notice that the time integral of the parameter value itself, as argument of the function, account for the memory of the component
- The term $\bar{V}(\bar{\alpha}_{DS}, \bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t)$ is the rate of change. Here for a discontinuous parameter mathematically it should be defined as the value of the instantaneous $\bar{\alpha}_{DS}$ change

To be noticed that the history dependence introduced inside the probability \bar{p} make the system not Markowian. Note that the system can return to be Markovian by expanding the phase space (i.e., increase its dimensionality).

In this respect, we introduce a new dimension in the phase space: the time at which the parameters changed status and the correspondent value $\{(\bar{\alpha}_{DS}, t)_i\} = \{\bar{\alpha}_{DS_i}, t_i\} = \bar{\alpha}_{DS}, \bar{t}$ (for $i = 1, \dots, n$)

Eq. 17 assumes now the form:

$$\frac{\partial \bar{\alpha}_{DS}}{\partial t} = \bar{\delta}(\bar{\alpha}_{DS}, \bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t) \bar{V}(\bar{\alpha}_{DS}, \bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t) \bar{p}(\bar{\alpha}_{DS}, \bar{t}, \bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t)$$

for $t \geq t_n$

Eq. 18

This formulation introduce a phase space continuously growing over time that poses a challenge in term of being capable to define a priori the size of the problem and still represents a research topics that will impact future evolution of the RAVEN software design.

Several simplifications are possible:

1. if the memory of the past is not affected by the time distance then

$$\bar{p}(\bar{\alpha}_{DS}, \bar{t}, \bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brow}, t) = \bar{p}(\bar{\alpha}_{DS}, \bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brow}, t)$$

Eq. 19

2. if the number of samples is predictable before the simulation itself takes place in (e.g. n times), then the dimensionality of the phase space could be fixed and the different $\bar{\alpha}_{DS_i}$ could be treated explicitly as $\bar{\alpha}_{Staz}$ while \bar{t} (if the above simplification is not valid) should remain as a dimension of the phase space. In this case \bar{t} still needs to be computed and its expression is:

$$\bar{t}(t) = \int_{t_0}^t dt \bar{t} \delta(\bar{\alpha}_{DS}, \bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brow}, t)$$

Eq. 20

3. Another possible approximation alternative to the previous one is that the memory of system (here explicitly represented by $\int_{t_0}^t dt \bar{\alpha}_{DS}$) is limited only to a fix number of step back in the past, in this case n is always bounded. Therefore adding $\{\bar{\alpha}_{DS_i}, t_i\}$ (for $i=1, \dots, n$) keep the system Markovian. Unfortunately this situation is deeply different from approximation 2 where the fact that $\bar{\alpha}_{DS_i}$ do not change over time allow them to be treated as $\bar{\alpha}_{Staz}$ and so possible excluded from the space phase. From the simulation point of view that imply that this probability checks need to be performed during the simulation itself.

One of the most interesting cases from point of view of the analysis that RAVEN will be performing is a combination of approximation 1 and 3. Under such assumption the system is again Markovian once $\{\bar{\alpha}_{DS_i}\}$ (for $i=1, \dots, n$) is added to the phase space. An example of this case is a component whose probability to correctly act on demand depends on his failure/success at the last demand.

When we combine approximation 1 and 3 we obtain the physical model corresponding to discrete jumps or collision. In fact once the phase space contains this additional variable the equation describing the trajectory of the systems along those dimensions present a discontinuity given either by the presence of an Heaviside function or by the speed function that could eventually be a distribution function.

An example of this type of modeling is the scattering kernel of the transport equation where neutron suddenly change direction with a probability (memory-less) by unit of length (in our case in time) given by the cross sections.

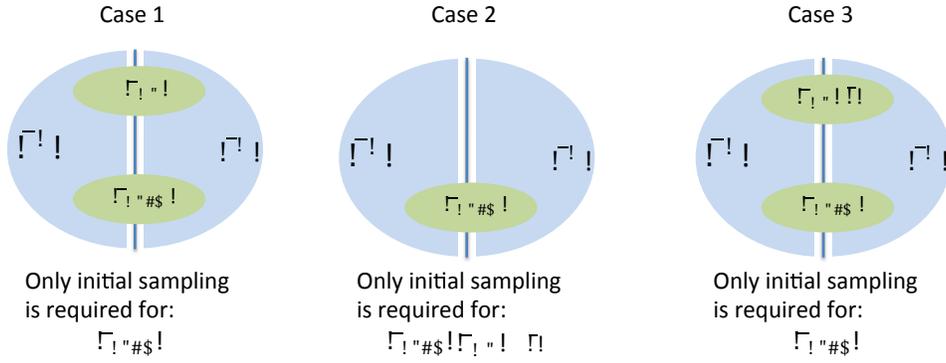


Figure 4: Phase space structure for three different cases: simplification 1 is valid (case 1), simplification 2 is valid (case 2), simplification 3 is valid (case 3)

Figure 4 give an overview of the structure of the phase space for the three different simplifications:

- Simplification 1: $\bar{\alpha}_{DS} = \bar{\alpha}_{DS}^c \cup \bar{\alpha}_{DS}^d$ where $\begin{cases} \bar{\alpha}_{DS}^c \subset \bar{\theta}^c \\ \bar{\alpha}_{DS}^d \subset \bar{\theta}^d \end{cases}$ and $\bar{\alpha}_{Brow} = \bar{\alpha}_{Brow}^c \cup \bar{\alpha}_{Brow}^d$ where $\begin{cases} \bar{\alpha}_{Brow}^c \subset \bar{\theta}^c \\ \bar{\alpha}_{Brow}^d \subset \bar{\theta}^d \end{cases}$
- Simplification 2: $\bar{\alpha}_{Brow} = \bar{\alpha}_{Brow}^c \cup \bar{\alpha}_{Brow}^d$ where $\begin{cases} \bar{\alpha}_{Brow}^c \subset \bar{\theta}^c \\ \bar{\alpha}_{Brow}^d \subset \bar{\theta}^d \end{cases}$
- Simplification 3: $\bar{\alpha}_{DS} = \bar{\alpha}_{DS}^c \cup \bar{\alpha}_{DS}^d$ where $\begin{cases} \bar{\alpha}_{DS}^c \subset \bar{\theta}^c \\ \bar{\alpha}_{DS}^d \subset \bar{\theta}^d \end{cases}$, $\bar{\alpha}_{Brow} = \bar{\alpha}_{Brow}^c \cup \bar{\alpha}_{Brow}^d$ where $\begin{cases} \bar{\alpha}_{Brow}^c \subset \bar{\theta}^c \\ \bar{\alpha}_{Brow}^d \subset \bar{\theta}^d \end{cases}$ and $\bar{t} = \bar{t}^c \cup \bar{t}^d$ where $\begin{cases} \bar{t}^c \subset \bar{\theta}^c \\ \bar{t}^d \subset \bar{\theta}^d \end{cases}$

The system Eq. 16, accounting also for $\bar{\alpha}_{DS}$ takes the form:

$$\left\{ \begin{array}{l} \frac{\partial \bar{\theta}^c}{\partial t} = f(\bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brow}, t) \\ \frac{\partial \bar{\theta}^d}{\partial t} = g(\bar{\theta}^d, \bar{\theta}^c, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brow}, t) \\ \frac{\partial \bar{\alpha}_{Brow}}{\partial t} = b(\bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brow}, t) \Gamma(t) \\ \frac{\partial \bar{\alpha}_{DS}}{\partial t} = \delta(\bar{\alpha}_{DS}, \bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brow}, t) \cdot \\ \quad \cdot \bar{V}(\bar{\alpha}_{DS}, \bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brow}, t) \bar{p} \left(\int_{t_0}^t dt \bar{\alpha}_{DS}, \bar{\alpha}_{DS}, \bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brow}, t \right) \\ \Pi(\bar{\theta}^c, t_0) \sim pdf(\bar{\theta}_0^c | \sigma_c^2) \\ \Pi(\bar{\theta}^d, t_0) \sim pdf(\bar{\theta}_0^d | \sigma_d^2) \\ \bar{\alpha}_{Staz}(t) = \bar{\alpha}_{Staz}(t_0) \sim pdf(\alpha_{Staz}^0 | \sigma_{Staz}^2) \\ \bar{\alpha}_{DS}(t_0) = \alpha_{DS}^0 \\ \bar{\alpha}_{Brow}(t_0) \sim \alpha_{Brow}^0 \Gamma(t_0) \end{array} \right.$$

Eq. 21

By introduction of approximation 1 and 3:

$$\left\{ \begin{array}{l} \frac{\partial \bar{\theta}^c(t)}{\partial t} = f(\bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t) \\ \frac{\partial \bar{\theta}^d(t)}{\partial t} = g(\bar{\theta}^d, \bar{\theta}^c, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t) \\ \frac{\partial \bar{\alpha}_{Brown}}{\partial t} = b(\bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t) \Gamma(t) \\ \frac{\partial \bar{\alpha}_{DS}}{\partial t} = \bar{\delta}(\bar{\alpha}_{DS}, \bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t) \cdot \\ \quad \cdot \bar{V}(\bar{\alpha}_{DS}, \bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t) \bar{p}(\bar{\alpha}_{DS}, \bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t) \\ \Pi(\bar{\theta}^c, t_0) \sim pdf(\bar{\theta}_0^c |, \sigma_c^2) \\ \Pi(\bar{\theta}^d, t_0) \sim pdf(\bar{\theta}_0^d |, \sigma_d^2) \\ \bar{\alpha}_{Staz}(t) = \bar{\alpha}_{Staz}(t_0) \sim pdf(\alpha_{Staz}^0 |, \sigma_{Staz}^2) \\ \bar{\alpha}_{DS}(t_0) = \alpha_{DS}^0 \\ \bar{\alpha}_{Brown}(t_0) \sim \alpha_{Brown}^0 \Gamma(t_0) \end{array} \right.$$

Eq. 22

From a software implementation point of view it is important to distinguish explicitly what should be computed before the simulation starts and during the simulation itself. Starting from the above system this distinction has the following form:

$$\left\{ \begin{array}{l} \Pi(\bar{\theta}^c, t_0) \sim pdf(\bar{\theta}_0^c |, \sigma_c^2) \\ \Pi(\bar{\theta}^d, t_0) \sim pdf(\bar{\theta}_0^d |, \sigma_d^2) \\ \bar{\alpha}_{Staz}(t) = \bar{\alpha}_{Staz}(t_0) \sim pdf(\alpha_{Staz}^0 |, \sigma_{Staz}^2) \\ \bar{\alpha}(t_0) = \alpha_{DS}^0 \\ \bar{\alpha}_{Brown}(t_0) \sim \alpha_{Brown}^0 \Gamma(t_0) \end{array} \right.$$

Eq. 23

$$\left\{ \begin{array}{l} \frac{\partial \bar{\theta}^c(t)}{\partial t} = f(\bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t) \\ \frac{\partial \bar{\theta}^d(t)}{\partial t} = g(\bar{\theta}^d, \bar{\theta}^c, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t) \\ \frac{\partial \bar{\alpha}_{Brown}}{\partial t} = b(\bar{\theta}^c, \bar{\theta}^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t) \Gamma(t) \end{array} \right.$$

Eq. 24

Of course this discussion did not cover all the possible phenomena, for example cases where the structure of the system of equation changes in time, but probably provide a sufficient framework to extrapolate to the cases here not explicitly treated.

Given the presence of all these sources of probabilistic behavior every simulation performed by RAVEN/RELAP-7 is nothing more than one of the possible path of the system in the phase space. In such a case rather than one of the possible solution is much more informative the probability of this solution to really take place. The difference in the trajectory in the phase space between a deterministic and probabilistic system is depicted in Figure 5 where $\Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0)$ is the probability distribution function and the $\mathcal{L}_{CK}^c(\Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0))$ the Chapman-Kolmogorov operator. The explanation of those concepts is matter of the next chapter.

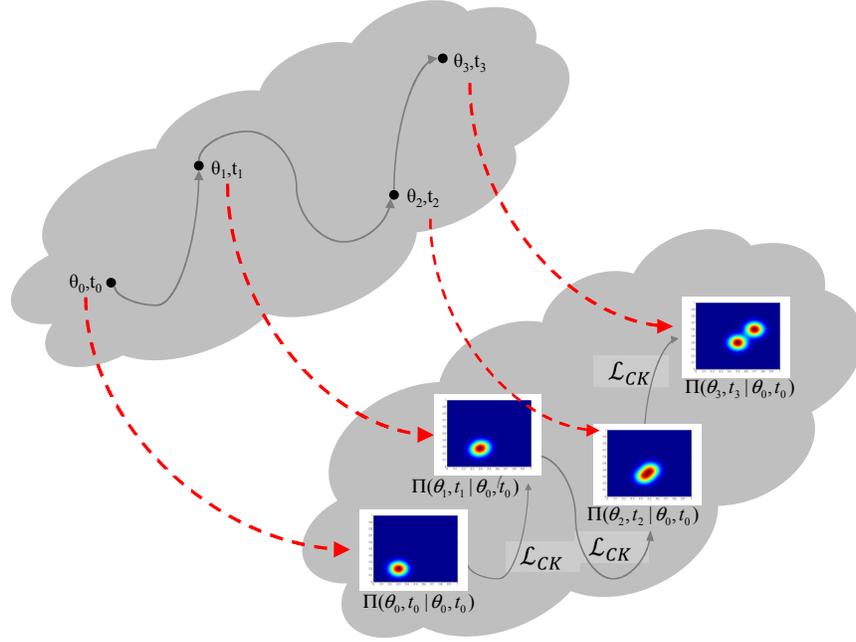


Figure 5: Deterministic (top) vs Probabilistic (bottom) analysis of system dynamics

3.1.2 Formulation of the equation set in a statistical framework

Given the above mentioned premises and assuming that at least one of the simplification considerate in the previous chapter is applicable (so that the system could be casted as Markovian) it is natural to investigate the evolution of the system in terms of its probability density function in the global phase space $\bar{\theta}$ via the Chapman-Kolmogorov equation [6,7].

The integral form of the Chapman-Kolmogorov is the following:

$$\Pi(\bar{\theta}_3, t_3 | \bar{\theta}_1, t_1) = \int d\bar{\theta}_2 \Pi(\bar{\theta}_2, t_2 | \bar{\theta}_1, t_1) \Pi(\bar{\theta}_3, t_3 | \bar{\theta}_2, t_2) \quad \text{where } t_1 < t_2 < t_3$$

Eq. 25

while its differential form is:

$$\frac{\partial \Pi(\bar{\theta}, t | \bar{\theta}_0, t_0)}{\partial t} = \mathcal{L}_{CK}(\Pi(\bar{\theta}, t | \bar{\theta}_0, t_0))$$

Eq. 26

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

$$\lim_{\Delta t \rightarrow 0} \frac{1}{\Delta t} \int_{|\bar{\theta}_2 - \bar{\theta}_1| < \varepsilon} \Pi(\bar{\theta}_2, t + \Delta t | \bar{\theta}_1, t) d\bar{\theta}_2 = 0$$

Eq. 27

$$\lim_{\Delta t \rightarrow 0} \frac{1}{\Delta t} \Pi(\bar{\theta}_2, t + \Delta t | \bar{\theta}_1, t) = W(\bar{\theta}_2 | \bar{\theta}_1, t)$$

Eq. 28

$$\lim_{\Delta t \rightarrow 0} \frac{1}{\Delta t} \int_{|\bar{\theta}_2 - \bar{\theta}_1| < \varepsilon} (\theta_{2,i} - \theta_{1,i}) \Pi(\bar{\theta}_2, t + \Delta t | \bar{\theta}_1, t) d\bar{\theta}_2 = A_i(\bar{\theta}_1, t) + \mathcal{O}(\varepsilon)$$

Eq. 29

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

Eq. 30

The first condition guarantees the continuity of $\Pi(\bar{\theta}, t | \bar{\theta}_0, t_0)$, while the other three force the finite existence of three parameters that will be described in Section 8.1.3.

Equation 25 can be furthermore decomposed into the continuous and discrete components:

$$\begin{cases} \Pi(\bar{\theta}_3^c, t_3 | \bar{\theta}_1^c, t_1) = \int d\bar{\theta}_2^c \Pi(\bar{\theta}_2^c, t_2 | \bar{\theta}_1^c, t_1) \Pi(\bar{\theta}_3^c, t_3 | \bar{\theta}_2^c, t_2) \\ \Pi(\bar{\theta}_3^d, t_3 | \bar{\theta}_1^d, t_1) = \int d\bar{\theta}_2^d \Pi(\bar{\theta}_2^d, t_2 | \bar{\theta}_1^d, t_1) \Pi(\bar{\theta}_3^d, t_3 | \bar{\theta}_2^d, t_2) \end{cases} \quad \text{where } t_1 < t_2 < t_3$$

Eq. 31

and its differential form is as follows:

$$\begin{cases} \frac{\partial \Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0)}{\partial t} = \mathcal{L}_{CK}^c(\Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0), \bar{\theta}^d, \bar{\alpha}_{Brow}, \bar{\alpha}_{Staz}, \bar{\alpha}_{DS}, t) \\ \frac{\partial \Pi(\bar{\theta}^d, t | \bar{\theta}_0^d, t_0)}{\partial t} = \mathcal{L}_{CK}^d(\Pi(\bar{\theta}^d, t | \bar{\theta}_0^d, t_0), \bar{\theta}^c, t) \end{cases}$$

Eq. 32

where:

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

3.1.3 The Chapman-Kolmogorov Equation

As mentioned in the previous paragraph, the system of equations (2) which is written in integral form can be solved in a differential form through the Chapman-Kolmogorov (C-K) operator [6, 7]:

$$\begin{aligned} \frac{\partial \Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0)}{\partial t} = & - \sum_i \frac{\partial}{\partial \theta_i^c} (A_i(\bar{\theta}^c, \bar{\theta}^d, t) \Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0)) + \frac{1}{2} \sum_{i,j} \frac{\partial^2}{\partial \theta_i^c \partial \theta_j^c} (B_{i,j}(\bar{\theta}^c, \bar{\theta}^d, t) \Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0)) \\ & + \int (W(\bar{\theta}^c | \bar{\theta}'^c, \bar{\theta}^d, t) \Pi(\bar{\theta}'^c, t | \bar{\theta}_0^c, t_0) - W(\bar{\theta}'^c | \bar{\theta}^c, \bar{\theta}^d, t) \Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0)) d\bar{\theta}'^c \end{aligned}$$

Eq. 33

$$\frac{\partial \Pi(\bar{\theta}^d, t | \bar{\theta}_0^d, t_0)}{\partial t} = \sum_i W(\bar{\theta}^d | \bar{\theta}_i^d, \bar{\theta}^c, t) \Pi(\bar{\theta}_i^d, t | \bar{\theta}^d, t_0) - W(\bar{\theta}_i^d | \bar{\theta}^d, \bar{\theta}^c, t) \Pi(\bar{\theta}^d, t | \bar{\theta}_0^d, t_0)$$

Eq. 34

where

$$A_i(\bar{\theta}, t) = \begin{cases} 0 & \text{if } \bar{\theta}_i \in \bar{\theta}^d \\ f(\bar{\theta}^c, \bar{\theta}^d, \alpha_{Brow}, \alpha_{Staz}, t) + \frac{1}{2} \frac{\partial b(\bar{\theta}^c, t)}{\partial \bar{\theta}^c} Q b(\bar{\theta}^c, t) & \text{otherwise} \end{cases}$$

$$B_{i,j}(\bar{\theta}, t) = \begin{cases} 0 & \text{if } \bar{\theta}_i \text{ or } \bar{\theta}_j \in \bar{\theta}^d \\ b(\bar{\theta}^c, t) Q b^T(\bar{\theta}^c, t) & \text{otherwise} \end{cases}$$

Eq. 35

This system of equations is composed of four parts that identify four different types of process known as drift, diffusion, jumps and component state transitions. These four processes are described in Sections 2.1 through 2.4.

3.1.3.1 Drift

The drift process is described by the Liouville's equation:

$$\frac{\partial \Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0)}{\partial t} = \sum_i \frac{\partial}{\partial \theta_i^c} (A_i(\bar{\theta}^c, \bar{\theta}^d, t) \Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0))$$

Eq. 36

Note that this equation describes a completely deterministic motion indicated by the equation:

$$\frac{\partial \bar{\theta}^c(t)}{\partial t} = A(\bar{\theta}^c, \bar{\theta}^d, t)$$

If $\bar{\theta}^c(\bar{\theta}_0^c, \bar{\theta}^d, t)$ is the solution of $\frac{\partial \bar{\theta}^c(t)}{\partial t} = A(\bar{\theta}^c, \bar{\theta}^d, t)$, then the solution of the Liouville equation is:

$$\Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0) = \delta(\bar{\theta}^c - \bar{\theta}^c(\bar{\theta}_0^c, \bar{\theta}^d, t))$$

provided the initial condition

$$\Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0) = \delta(\bar{\theta}^c - \bar{\theta}_0^c)$$

3.1.3.2 Diffusion

This process is described by the Fokker-Plank equation:

$$\frac{\partial \Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0)}{\partial t} = - \sum_i \frac{\partial}{\partial \theta_i^c} (A_i(\bar{\theta}^c, \bar{\theta}^d, t) \Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0)) + \frac{1}{2} \sum_{i,j} \frac{\partial^2}{\partial \theta_i^c \partial \theta_j^c} (B_{i,j}(\bar{\theta}^c, \bar{\theta}^d, t) \Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0))$$

Eq. 37

where $A(\bar{\theta}^c, \bar{\theta}^d, t)$ is the drift vector and $B_{i,j}(\bar{\theta}^c, \bar{\theta}^d, t)$ is the diffusion matrix. Provided the initial condition:

$$\Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0) = \delta(\bar{\theta}^c - \bar{\theta}_0^c)$$

the Fokker-Plank equation describes a system moving with drift whose velocity is $A(\bar{\theta}^c, \bar{\theta}^d, t)$ on which is superimposed a Gaussian fluctuation with covariance matrix $B(\bar{\theta}^c, \bar{\theta}^d, t)$.

3.1.3.3 Jumps in continuous space

This process is described by the Master equation:

$$\frac{\partial \Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0)}{\partial t} = \int (W(\bar{\theta}^c | \bar{\theta}'^c, \bar{\theta}^d, t) \Pi(\bar{\theta}'^c, t | \bar{\theta}_0^c, t_0) - W(\bar{\theta}'^c | \bar{\theta}^c, \bar{\theta}^d, t) \Pi(\bar{\theta}^c, t | \bar{\theta}_0^c, t_0)) d\bar{\theta}'^c$$

Eq. 38

and, provided the initial condition

$$\Pi(\bar{\theta}^c, t_0 | \bar{\theta}_0^c, t_0) = \delta(\bar{\theta}^c - \bar{\theta}_0^c)$$

it describes a process characterized by straight lines interspersed with discontinuous jumps whose distribution is given by $W(\bar{\theta}^c | \bar{\theta}'^c, \bar{\theta}^d, t)$.

3.1.3.4 Jumps in discrete space

Transitions in the discrete space can occur in terms of jumps, then the formulation of $\frac{\partial \Pi(\bar{\theta}^d, t | \bar{\theta}_0^d, t_0)}{\partial t} = \mathcal{L}_{CK}^d(\Pi(\bar{\theta}^d, t | \bar{\theta}_0^d, t_0), \bar{\theta}^c, t)$ is similar to the Master equation rewritten for a discrete phase space:

$$\frac{\partial \Pi(\bar{\theta}^d, t | \bar{\theta}_0^d, t_0)}{\partial t} = \sum_i W(\bar{\theta}^d | \bar{\theta}_i^d, \bar{\theta}^c, t) \Pi(\bar{\theta}_i^d, t | \bar{\theta}_0^d, t_0) - W(\bar{\theta}_i^d | \bar{\theta}^d, \bar{\theta}^c, t) \Pi(\bar{\theta}^d, t | \bar{\theta}_0^d, t_0)$$

Eq. 39

Note that the similarity with the equation that describes temporal evolution of multi-state Markov model:

- The first term $W(\bar{\theta}^d | \bar{\theta}_i^d, \bar{\theta}^c, t)$ indicates the transition probability from $\bar{\theta}_i^d$ to $\bar{\theta}^d$
- The second term $W(\bar{\theta}_i^d | \bar{\theta}^d, \bar{\theta}^c, t)$ indicates the transition probability from $\bar{\theta}^d$ to $\bar{\theta}_i^d$.

3.2 RAVEN solution algorithms

Analysis of stochastic systems can be extremely challenging due to the complexity and high dimensionality of the Chapman-Kolmogorov equation. An analytical solution is available only for very simple cases. When an analytical solution is not available, numerical methods are often employed. In this respect, two different approaches can be followed to solve Equations 33 and 34:

1. Determine approximate solutions of the exact problems
2. Determine the exact solution for the approximate models

•

Due to the very large complexity and the high dimensionality of the system considered, RAVEN will follow the first approach. In particular, two algorithms will be employed:

1. Monte-Carlo
2. Dynamic Event Tree

•

The main idea is to run a set simulation runs having different set of $\bar{\alpha}_{Brow}$, $\bar{\alpha}_{Staz}$, $\bar{\alpha}_{DS}$, \bar{t} and initial conditions and terminate them until one of the following stopping conditions is reached:

- Mission time is reached (i.e., a user specified value of time)
- Top event is reached (e.g., maximum temperature of the clad or core damage)

3.2.1 Monte-Carlo

Given:

$$\frac{\partial \bar{\theta}^c(t)}{\partial t} = f(\bar{\theta}^c, \bar{\theta}_i^d, \bar{\alpha}_{Staz}, \bar{\alpha}_{Brown}, t)$$

lets define a function g_i which represents the solution the previous equation:

$$\bar{\theta}^c(t) = g_i(t, \bar{\theta}_0^c)$$

i.e., g_i represents the trajectory in the $\bar{\theta}^c$ space for a fixed $\bar{\theta}_i^d$ and an initial condition $\bar{\theta}_0^c$.

Analysis of dynamic stochastic systems through Monte-Carlo algorithms is performed as following (see Fig.5):

1. Sample:
 - i) A value for $\bar{\alpha}_{Staz}, \bar{\alpha}_{DS}, \bar{t}$ depending on which approximations are valid
 - ii) The initial conditions, i.e., $\bar{\theta}_0^c, \bar{\theta}_0^d$ for $t_0 = 0$
 - iii) Transition conditions, i.e., from $W(\bar{\theta}^d | \bar{\theta}_i^d, \bar{\theta}^c, t)$ sample points in the state space where a transition from $\bar{\theta}_i^d$ to $\bar{\theta}_j^d \forall i, j$ occurs
2. Run the system simulator using the values previously sampled and change $\bar{\alpha}_{Brown}$ (and $\bar{\alpha}_{DS}, \bar{t}$ when possible) at each time step.
3. Stop the simulation (e.g., at time t_1) when a transition condition is reached (see step 1.III), and move from the actual $\bar{\theta}_0^d$ to the new $\bar{\theta}^d$ (e.g., $\bar{\theta}_1^d$)
4. Run the simulation as performed in step 3 starting from the new coordinate $(\bar{\theta}_1^c = g_0(t_1, \bar{\theta}_0^c), \bar{\theta}_1^d, t_1)$ and stop when a new transition condition is reached.
5. Repeat steps 3 and 4 until a stopping condition is reached
6. Repeat 1 through 4 for a large number of runs set by the user.

$$\begin{aligned} & (\bar{\theta}_0^c, \bar{\theta}_0^d, t_0) \xrightarrow{f(\bar{\theta}^c, \bar{\theta}_0^d, \alpha_{Brown}, \alpha_{Staz}, t)} (\bar{\theta}_1^c = g_0(t_1, \bar{\theta}_0^c), \bar{\theta}_0^d, t_1) \xrightarrow{(\bar{\theta}_0^d \rightarrow \bar{\theta}_1^d)} \\ \Rightarrow & (\bar{\theta}_1^c, \bar{\theta}_1^d, t_1) \xrightarrow{f(\bar{\theta}^c, \bar{\theta}_1^d, \alpha_{Brown}, \alpha_{Staz}, t)} (\bar{\theta}_2^c = g_1(t_2, \bar{\theta}_1^c), \bar{\theta}_1^d, t_2) \xrightarrow{(\bar{\theta}_1^d \rightarrow \bar{\theta}_2^d)} \\ \Rightarrow & (\bar{\theta}_2^c, \bar{\theta}_2^d, t_2) \xrightarrow{f(\bar{\theta}^c, \bar{\theta}_2^d, \alpha_{Brown}, \alpha_{Staz}, t)} (\bar{\theta}_3^c = g_2(t_3, \bar{\theta}_2^c), \bar{\theta}_2^d, t_3) \xrightarrow{(\bar{\theta}_2^d \rightarrow \bar{\theta}_3^d)} \dots \end{aligned}$$

Figure 6: Example of Monte-Carlo run performed by RAVEN.

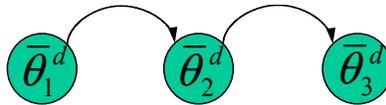


Figure 7: Example of 3-states system

An example is given for a 3 states system (see Fig.6) where:

- Transition from $\bar{\theta}_1^d$ to $\bar{\theta}_2^d$ depends on $\bar{\theta}^c$ and it is described by a cumulative distribution function shown in Fig.7 (left).
- Transition from $\bar{\theta}_2^d$ to $\bar{\theta}_3^d$ depends on the time spent in state $\bar{\theta}_2^d$ and it is described by a cumulative distribution function shown in Fig.7 (right).

Before each simulation run, the transition conditions from state 1 to state 2 and from state 2 to state 3 are determined by [9]:

- generating two random numbers (i.e., RNG_1 and RNG_2) in the interval $[0,1]$, and,
- determining the $\bar{\theta}^c = CDF_{12}^{-1}(RNG_1)$ and $\Delta t = CDF_{23}^{-1}(RNG_2)$

During the simulation run, when $\bar{\theta}^c = \bar{\theta}^c$ the transition from $\bar{\theta}_1^d$ to $\bar{\theta}_2^d$ occurs and after Δt , the transition from $\bar{\theta}_2^d$ to $\bar{\theta}_3^d$ takes place as shown in Fig.8.

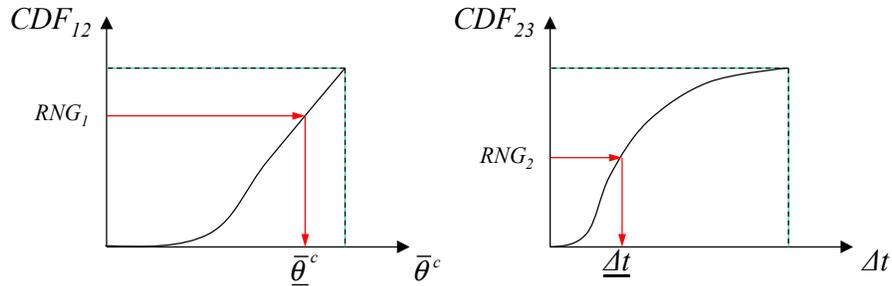


Figure 8: Examples of sampling of transition conditions from CDF for a 3-state system (see Fig. 6)

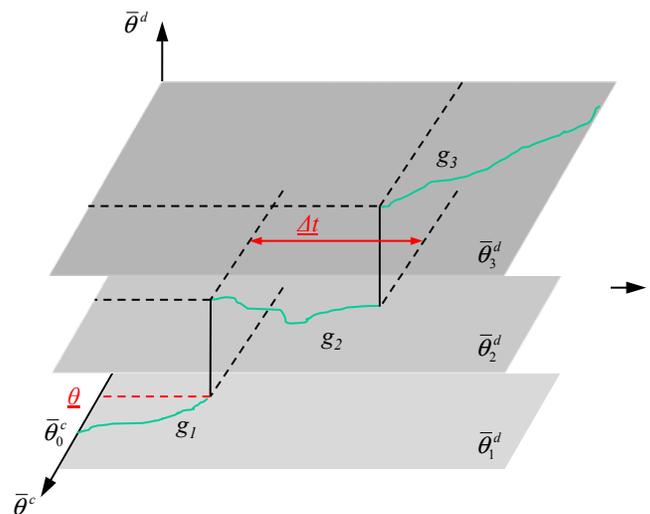


Figure 9: Example of Monte-Carlo analysis

3.2.1.1 Dynamic Event Tree

The concept of dynamic event trees can be summarized as the merging of:

- Classical Static Event Trees along with
- Monte-Carlo Analysis described in the previous section.
-

The classical (static) Event Tree (ET) methodology is used to determine those event sequences which can lead to an undesired state (i.e., any system failure such as core damage or radioactive release outside the containment) given an initiating event (i.e., a perturbation of the nominal conditions of the plant such as loss of off-site power).

An example of simple Event Tree is pictured in Figure 10, after the initiating event (i.e., a Large LOCA), the system begins to evolve until the point where a set point for the activation of a safety system is crossed. This may include the Reactor Trip system, Emergency Core Cooling System (ECCS), pressurizer relief valves, steam generator valves, etc.

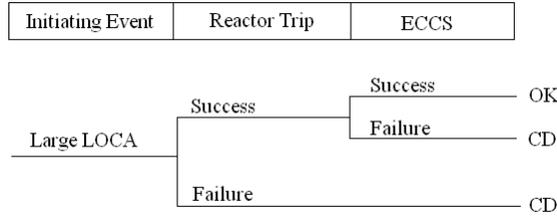


Figure 10: Example of event tree.

Each pathway on the event tree is considered to be one scenario of accident evolution. A sequence of successes or failures of the called-upon components gives one event tree scenario that may occur in a particular transient. Using the probabilities of failure for each of the called-upon systems, the probability of each scenario can be computed. Each of these scenarios leads to what is known as an end-state. Given the knowledge of what systems must actuate in order to prevent system failure within a certain period of time, the end-states can be labeled as either leading to core damage or to core-safe states.

However the classical Event-Tree methodology contains certain drawbacks that do not always allow for an appropriate modeling of system risk. One of the most pivotal drawbacks of conventional PRA methods is that time is not explicitly accounted for. During the course of an accident the exact timing of events could be important in scenario evolution especially when operator action and certain severe accident processes are considered (20). In addition, the methods of classical PRA does not always provide for mechanistic modeling of all systems and processes in a physically consistent manner. For these reasons, dynamic event tree (DET) methods have been developed over the past decades to overcome the deficiencies in conventional PRA modeling by merging the hierarchical structure of classical event trees and system simulators.

Conceptually, a DET algorithm is very similar to the Monte-Carlo algorithm presented in the previous section. The major differences are:

- the branching conditions (which correspond to the transition conditions in the Monte-Carlo algorithm) are fixed by the user and are not randomly sampled
- starting from an initial point in the state space, the algorithm generates new branches each time a branching condition is reached

The tree-like structure of the simulation allows to save computational time by not repeating, for each branch, the simulation performed for the all its parent branches.

A typical DET simulation is performed as following given the initial conditions and the set of branching conditions:

1. Sample:

- a. A value for $\bar{\alpha}_{Staz}, \bar{\alpha}_{DS}, \bar{t}$ depending on which approximations are valid
- b. The initial conditions, i.e., $\bar{\theta}_0^c, \bar{\theta}_0^d$ for $t_0 = 0$
- c. Transition conditions, i.e., from $W(\bar{\theta}^d | \bar{\theta}_i^d, \bar{\theta}^c, t)$ sample points in the state space where a transition from $\bar{\theta}_i^d$ to $\bar{\theta}_j^d \forall i, j$ occurs

2. Run the system simulator using the values previously sampled and change $\bar{\alpha}_{Brow}$ (and $\bar{\alpha}_{DS}, \bar{t}$ when possible) at each time step.
3. When a branching conditions is reached stop the simulation and generate a set of new branches accordingly to the branching conditions
4. Perform the simulation for each branch
5. Repeat 3 and 4 for each branch until a stopping condition is reached (e.g., simulation mission time is reached or a top event is reached)
6. Repeat 1 though 5 for a different set of initial conditions

An example of branching is given in Figure 11: starting from $(\bar{\theta}_0^c, \bar{\theta}_0^d, t_0)$ the simulation is performed by the system simulator until it reaches a branching condition at time t_1 . Accordingly to the branching conditions, a set of branches (usually two branches: event occurs or event does not occur) is generated. Each branch starts from the same initial conditions $\bar{\theta}_1^c = g_0(t_1, \bar{\theta}_0^c)$ at time t_1 and its corresponding $\bar{\theta}^d$.

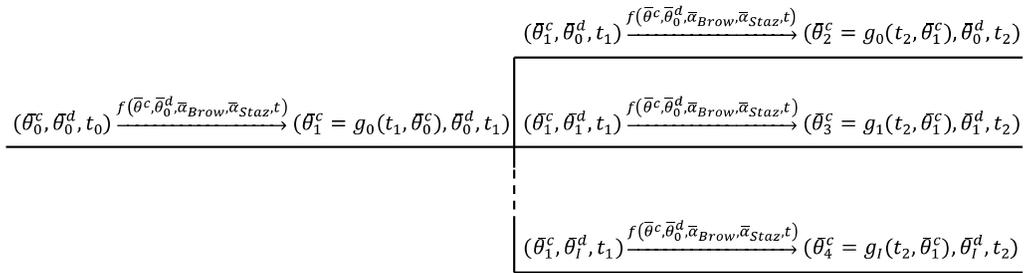


Figure 11: Example of DET branching

Two types of branching are considered here:

- At fixed time intervals: the mission time is divided into intervals and at the end of each time interval a branching occurs. Every possible branch is taken into account.
- At fixed points in the state space: branching occurs at specific points of the state space, when one of these points is reached in the simulation, a set of branches specific for that branching condition is generated.

Figures 10 through 13 show examples of these branching at fixed time intervals and at fixed point in the state space (Figure 13 and Figure 14) and a fixed time steps (Figure 15) for a 2-state model (Figure 12).

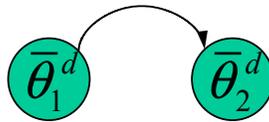


Figure 12: A 2-states Markov model

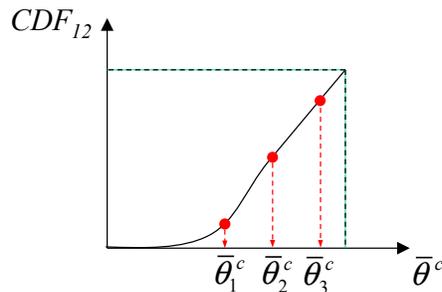


Figure 13: Set of branching conditions in the state space for a 2-state model

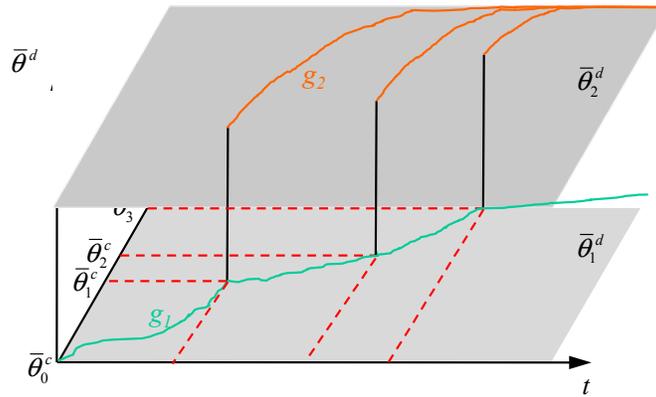


Figure 14: Example of branching at fixed points in the state space

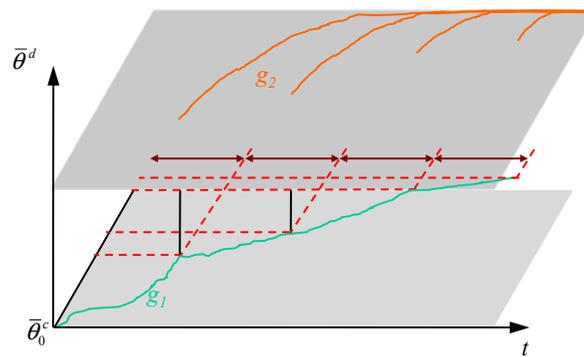


Figure 15: Example of branching at fixed time points

4 END OF THE YEAR ACCOMPLISHMENTS

4.1 Software Infrastructure

Following the schemes illustrated in the previous paragraphs the implementation of RAVEN has been started this current FY and moved forward quite fast.

Concerning the underlying software infrastructure (Figure 2) RAVEN is already functional. In the following the already available functionality are enlisted

- Input file is received and passed to RELAP-7 to set up the problem
- A grammar describing monitored, controlled, and auxiliary variables has been defined in a MOOSE-like structure. Below exemplification of the input structure is reported (explanatory comment are preceded by # sign).

INPUT	Description
<pre>[Controlled] control_logic_input = TMI_control_logic [./power_CH1] print_csv = true property_name = total_power_scaling data_type = double component_name = CH1 [./] [./]</pre>	<pre>#start of a controlled variable list #control logic file to be used #name of the controlled variable #printing comma separated values #which property is controlled (hard name) #data type #on which RELAP-7 component (soft name)</pre>

	#end of the controlled variable #end controlled variable list
[Monitored] ./max_temp_clad_CH1 operator = NodalMaxValue path = CLAD:TEMPERATURE data_type = double component_name = CH1 [./] [./]	#start of a monitored variable list #name of the monitored variable #restriction operator to be used $\bar{C} = \bar{G}(\bar{x}, t)$ #path and name of the monitored property #data type #on which RELAP-7 component (soft name) #end of the controlled variable #end monitored variable list
[RavenAuxiliary] ./scram_start_time data_type = double initial_value = 100000000 [./] [./]	#start of a auxiliary variable list #name of the auxiliary variable #data type #initial value #end of the auxiliary variable #end auxiliary variable list

- No new restriction $\bar{C} = \bar{G}(\bar{x}, t)$ operators have been implemented with respect the one already available via the MOOSE framework. Up to know the tested ones are:
 - ElementAverageValues: returns the average over the specified domain of one of the variables for which RELAP solve for (Temperature, Pressure, Velocity, Energy, Density)
 - NodalMaxValue: return the max of the nodal values assumed from a provided variable within a certain region of the computational domain
- All the variable so defined are made accessible to python by dictionaries constructing using SWIG (C++ python interface) in the user provided file [control logic file name.py]
- From the Python side the variable are accessible as class property in the following form:
 - auxiliary.[name]: provide access (read and write) to the auxiliary variable [name]
 - monitored.[name]: provide access (read) to the monitored variable [name]
 - controlled.[name]: provide access (read and write) to the controlled variable [name]
- After the initialization of the RELAP-7 solution a user defined function contained in the control logic file named: initial_function(monitored, controlled, auxiliary) is called to initialize the control logic variables. In case controlled variable are altered in this step the corresponding modification are carried inside the corresponding RELAP 7 equations
- RAVEN start then the simulation by calling RELAP-7
- At the end of each time step RAVEN evaluate a function named control_function(monitored, controlled, auxiliary) inside the control logic file where the user writes the control logic using the naming above defined

In appendix A it is showed the input file for a PWR based demo and the corresponding control logic.

4.2 GUI Current Capability

The GUI currently is aiding the graphical creation of the REALP 7 RAVEN input file, while still a lot need to be done to increase its friendliness and completeness, has its main software infrastructure completed.

The GUI is based on a software named Peacock that is a general GUI interface for MOOSE based application. The the GUI for RELAP-7 is rather different from the other MOOSE based application therefore a quite intensive work is needed to perform the specialization of the application and this done by taking advantage of API (Advanced Programming Interfaces) presents in Peacock.

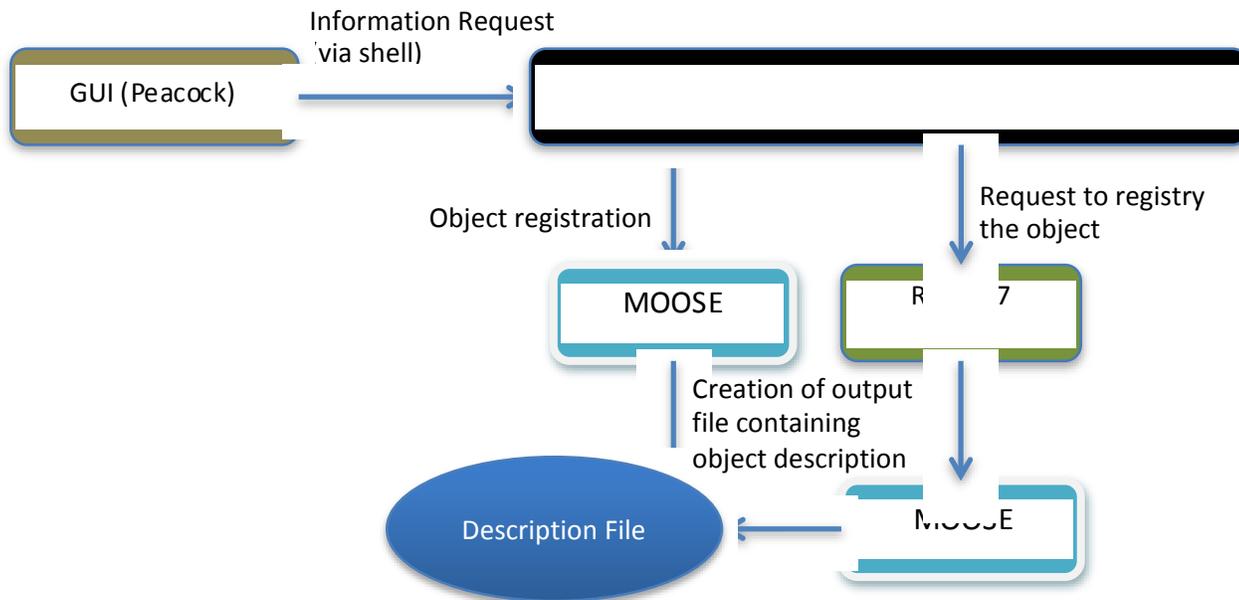


Figure 16: Information Flow for the Creation of the Input File GUI

4.2.1 Input/Visualization GUI Window

After this process is completed the GUI is able to aid the user to create an input file and a side windows will show what is currently present in the input file. The side window, which is a specific add on for the RAVEN project, not only show the ongoing creation of the plant layout but also allows inspecting the component already created and change the values of the describing parameters.

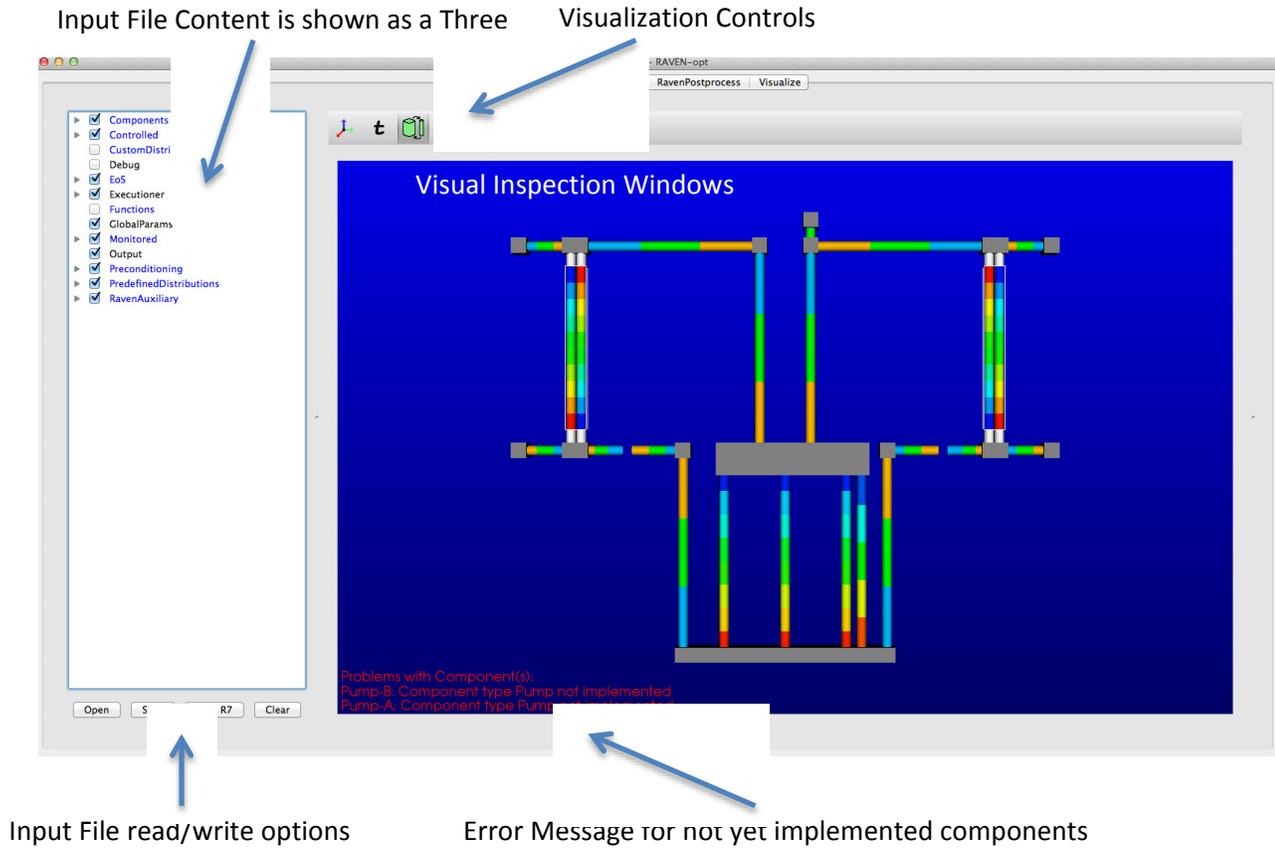


Figure 17: Input/plan Visualization GUI Window

4.2.2 The Execution Set Up Window

This window of the GUI has a central text box where the screen output of the program is shown while running and a set of buttons that will eventually open new windows to provide the setting information for the runs. Focusing on the specialization performed for RAVEN it is noticeable the specific windows to set up the multiple run used to perform Monte Carlo where the total number of sampling and maximum number of parallel sampling contemporaneously running is set. The window is showed in Figure 18.

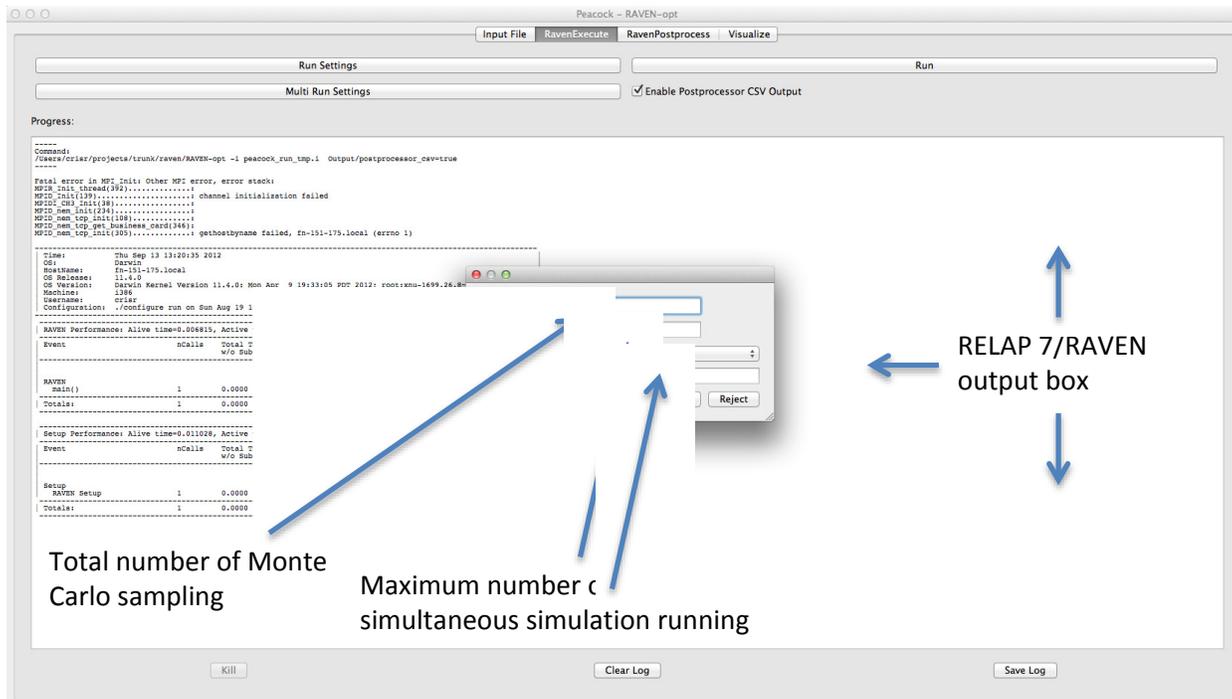


Figure 18: Running Control Windows

4.2.3 The Post Processor Visualization Window

The post processor window is used to inspect specific values in 2D plots **while running**. All the auxiliary, monitored, and controllable variables could be exported in comma separated values format files. This windows will inspect this file to find out which variable are present and allow the user from a drop down list to visualize them in 2D plots. Figure 19 show the window while operated by the user.

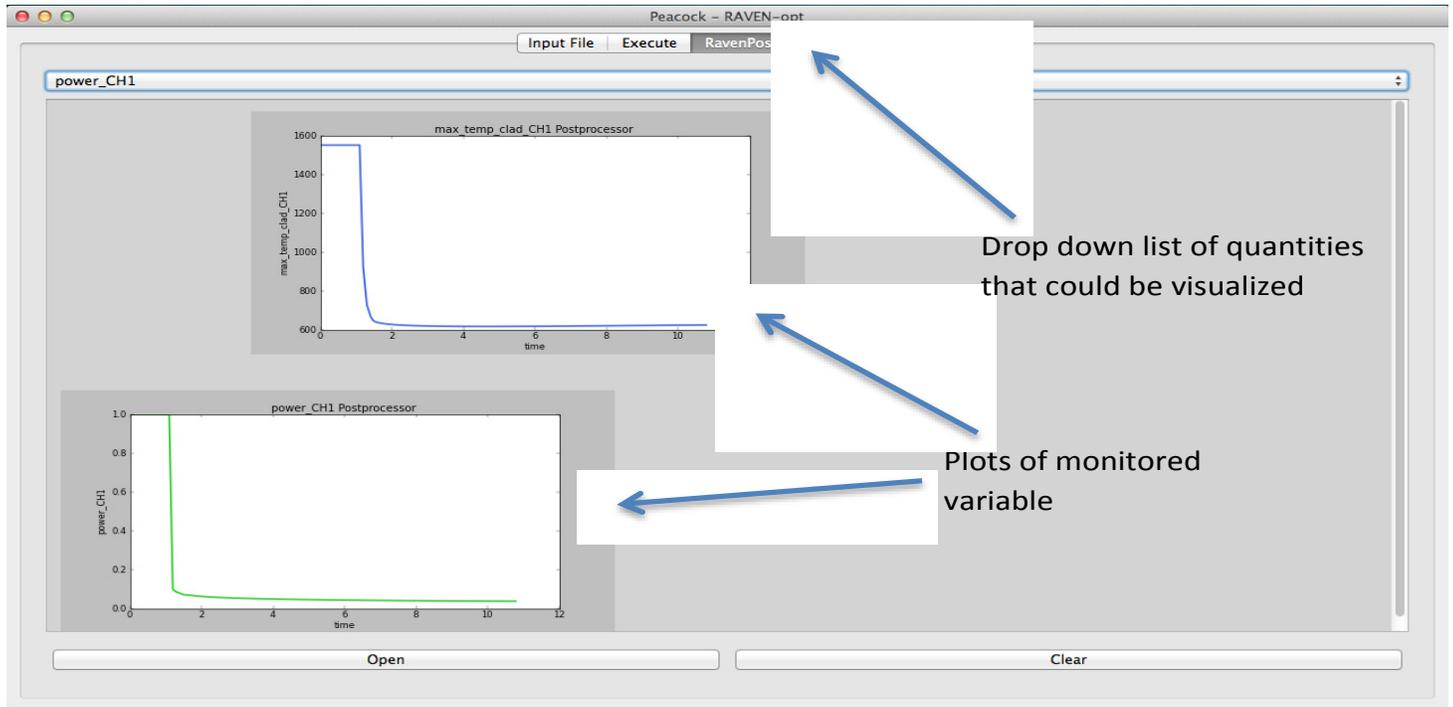


Figure 19: Post processor visualization windows

4.2.4 The Visualization Windows

For explanatory propose the visualization window can be imagined as project one of the solution field in the plant layout shown in the input windows. In reality it uses some features from the vtk graphical libraries to expand the one-dimensional solution to pipe like geometry and than it overlap the design from the input windows. In this window the user can follow the simulation by playing movies of the evolving solution. The quantities that is possible to visualize are all the ones for which REALP-7 solve for therefore there are some that are of numerical interested but also temperature, pressure, velocity, enthalpy and density.

Figure 20 shows the pressure status during a transient simulation.

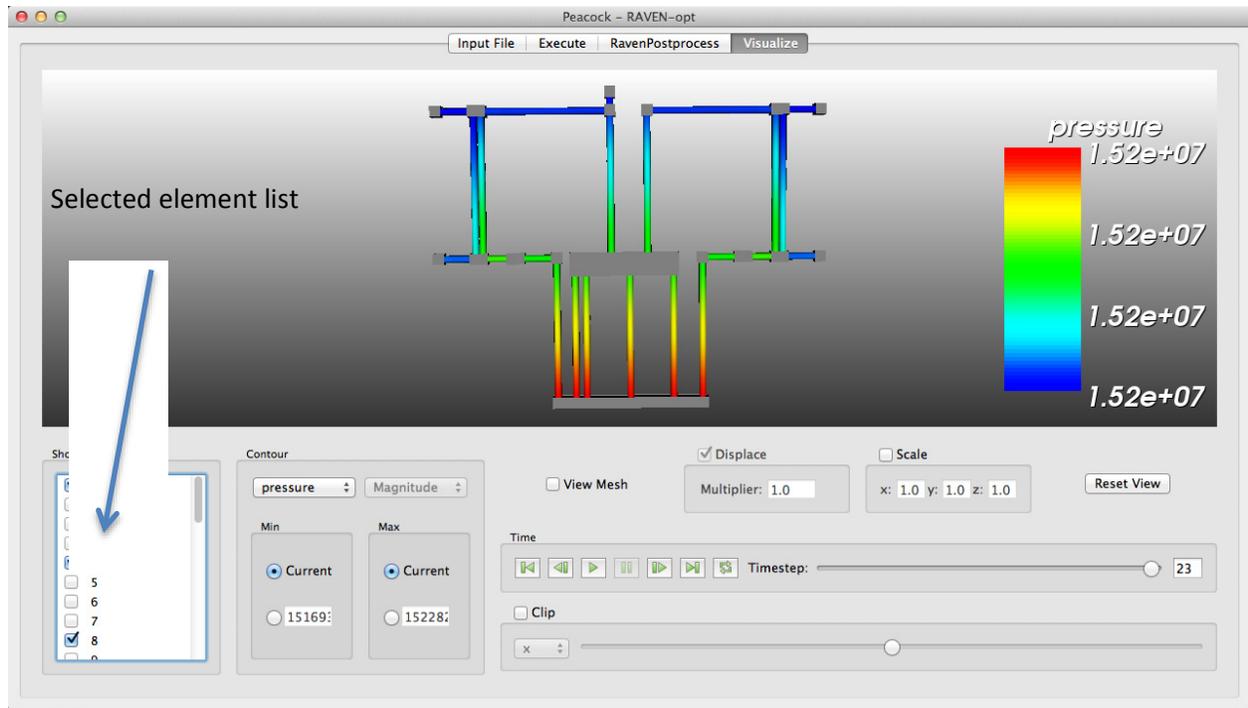


Figure 20 Movie visualization windows (solution while running)

4.3 Result From the Mathematical Framework

In this section two mathematical results will be shown to highlight the behaviors arising from different sources of stochastic behavior.

4.3.1 Harmonic oscillator

Let's consider a classical harmonic oscillator that can be described by the following equation:

$$\ddot{x}(t) + \omega^2 x(t) = 0 \quad 0 \leq t < \infty$$

Eq. 40

The coordinate in the phase space are given by:

$$\bar{X}(t) = \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix}$$

Eq. 41

That introduced in Eq. 41 lead to:

$$\dot{\bar{X}}(t) = \begin{bmatrix} 0 & 1 \\ \omega^2 & 0 \end{bmatrix} \bar{X}(t)$$

Eq. 42

The solution can be easily found as:

$$\bar{X}(t) = \begin{bmatrix} \cos \omega t & \frac{1}{\omega} \sin \omega t \\ -\omega \sin \omega t & \cos \omega t \end{bmatrix} \bar{X}(0)$$

Eq. 43

Assuming that the initial conditions are described by a normal distribution this example becomes illustrative of a probabilistic behavior driven from uncertainty on the initial conditions. Referring to the notation presented in the chapter on the mathematical framework the initial conditions (position and velocity) are two α_{staz} . The mathematical expression is:

$$\bar{X}(0) = \begin{bmatrix} x(0) \\ \dot{x}(0) \end{bmatrix} \sim \begin{bmatrix} N(0, \sigma_1) \\ N(0, \sigma_2) \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{1}{2}\left(\frac{x(0)}{\sigma_1}\right)^2} \\ \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{1}{2}\left(\frac{\dot{x}(0)}{\sigma_2}\right)^2} \end{bmatrix}$$

Eq. 44

The probability to find the system in a given point of the space phase is than (combined probability transformed into the product):

$$\Pi(x, t; \dot{x}, t) = \frac{1}{2\pi\sigma_1\sigma_2} e^{-\frac{1}{2}\left(\frac{x \cos \omega t - \frac{\dot{x}}{\omega} \sin \omega t}{\sigma_1} + \frac{-\omega x \sin \omega t + \dot{x} \cos \omega t}{\sigma_2}\right)^2}$$

Eq. 45

In the following picture we see the probability density function of the system in the phase space for at different instant.

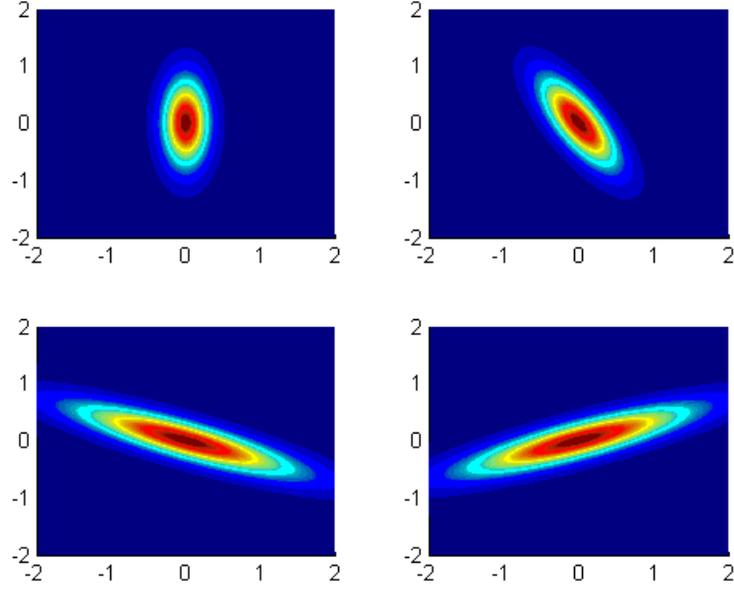


Figure 21: Harmonic oscillator: plot of $\Pi(x, t; \dot{x}, t)$ for different time steps

4.3.2 Cumulative Damage Time Evolution

The scope of this second example is to show a comparative analysis of the impact of a parameter subject to random distribution but not changing over time (α_{Staz}) or a parameter randomly changing as in Brownian motion like behavior (α_{Brow}). The system considered is a simplified model of material damage accumulation N . The starting equation is:

$$\begin{cases} \frac{dN}{dt} = \alpha \\ N(0) = N_0 \end{cases}$$

Eq. 46

Since α will assume a stochastic behavior it is more mathematically correct to use the integral from:

$$N(t) = \int_0^t dt\alpha + N_0$$

Eq. 47

Now two different cases are considered for the behavior of α :

$$\alpha = \begin{cases} \alpha_{Brow}(t) = N(\mu, \sigma) \\ \alpha_{Staz}(t) = \alpha_{Staz}(t=0) = N(\mu, \sigma) \end{cases}$$

Eq. 48

The probability distribution function r for N_{Brow} and N_{Staz} , respectively the solution for $\alpha = \alpha_{Brow}$ or $\alpha = \alpha_{Staz}$, are provided by the respective solution of:

$$\begin{cases} \frac{\partial r(N_{Brow}, t)}{\partial t} = -\alpha_{Brow} \frac{\partial r(N_{Brow}, t)}{\partial N} + \frac{\sigma}{2} \frac{\partial^2 r(N_{Brow}, t)}{\partial N^2} \\ r(N, 0) = \delta(N_{Brow} - N_0) \end{cases}$$

Eq. 49

$$\begin{cases} \frac{dN_{Staz}}{dt} = \alpha_{Staz} \\ N(0) = N_0 \end{cases}$$

Eq. 50

And the solutions are:

$$r(N_{Brow}, t) = \frac{1}{\sqrt{2\pi\sigma^2 t}} \exp\left[-\frac{(N - \mu t)^2}{2\sigma^2 t}\right]$$

Eq. 51

$$r(N_{Brow}, t) = \frac{1}{\sqrt{2\pi(\sigma t)^2}} \exp\left[-\frac{(N - \mu t)^2}{2(\sigma t)^2}\right]$$

Eq. 52

In the following Figure 22 the probability density function of the two different solutions are plotted (color scale) versus time and damage. Surprisingly the uncertainty is lower (lower dispersion) when modeling the speed of damage as a Brownian variable.

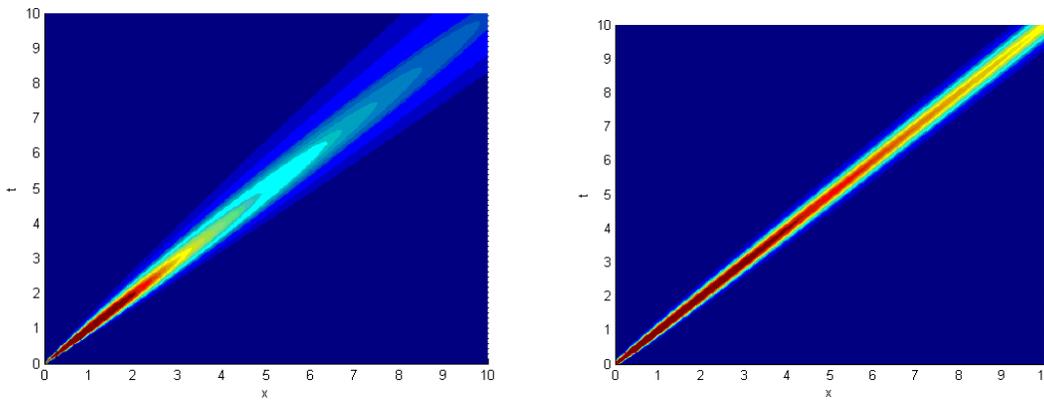


Figure 22: Uncertainty modeling of cumulative damage rate: \dot{N} constant in time (left) and continuously changing in time (right)

4.3.3 Combination of Control Logic and Stochastic Modeling

It is considered a pseudo transient where the simplified model in appendix A is used. The plant moves towards the steady state from an initial condition of not equilibrium. The list below summarizes the events that are controlled by RAVEN (in square brackets the corresponding type of RAVEN feature involved).

Problem set up:

- Monitored variables:
 - Average fuel temperature in the hot channel (CH1)
 - Average fuel temperature on the average channel (CH2)
- Controlled variables:
 - Thermal conductivity in channel CH1
 - Thermal conductivity in channel CH2

Initialization:

- The thermal conductivity in the gaps for both channels is sampled out from a normal distribution with mean value equal to the initial value and sigma of 5% [in this case the thermal conductivity is modeled as an α_{Staz}]

During simulation:

- When fuel average temperature reaches 500K the thermal conductivity in the gap is set equal to the thermal conductivity in the fuel [control on feedback]
- The thermal conductivity in the gap at each time step is sampled out of a normal distribution with mean value the thermal conductivity in the fuel and sigma of 5% [in this case the thermal conductivity is modeled as an α_{Brow}]

Monte Carlo sampling:

- 10 different simulation are simultaneous run to generate a distribution of the plan state

Figure 23 shows the comparison between transients where the control of the gap thermal conductivity is applied or not. To be noticed:

- The sudden spike in the clad temperature is due to the contact between fuel and clad that causes the thermal resistance to decrease
- The matching of the asymptotic values with or without the changes in the conductivity is due to the fact that the downstream thermal resistance has not changed

Figure 24 and Figure 25 plot, respectively the clad and fuel temperature profiles for 10 different runs where the thermal conductivity of the gap, both initial and after reaching 500K, is subject to a random distribution.

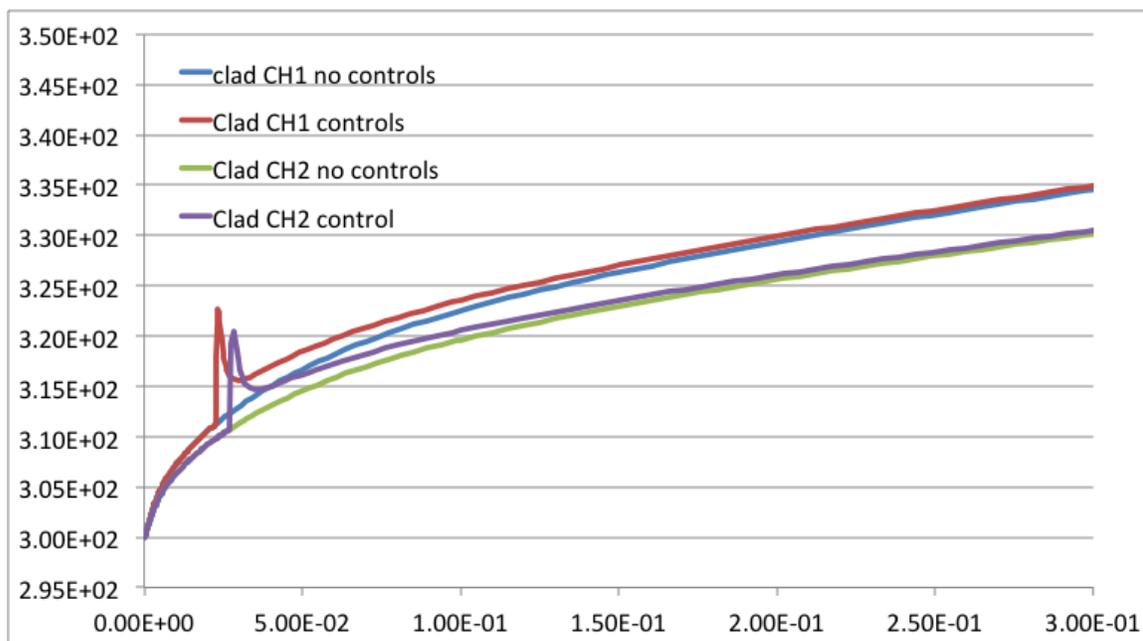


Figure 23: Comparison with or without the control logic acting on the gap thermal conductivity on the clad temperatures

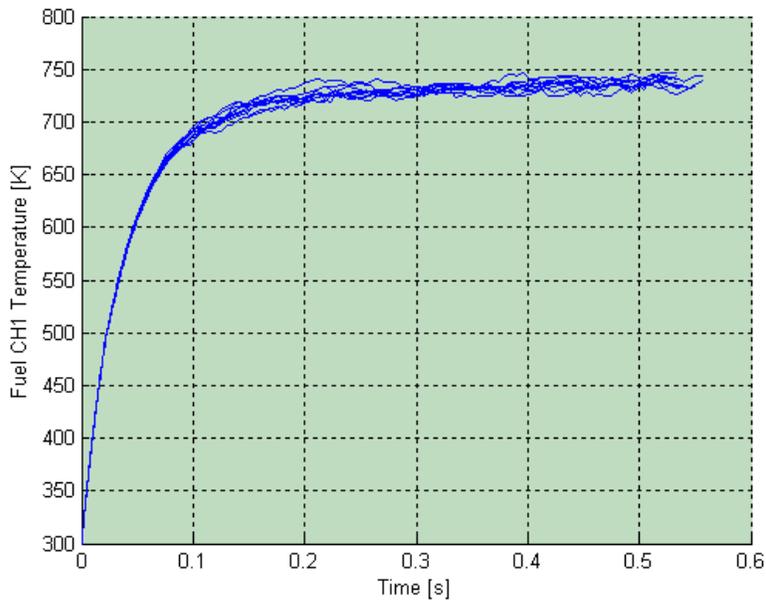


Figure 24: Fuel temperature behavior for the hot channel

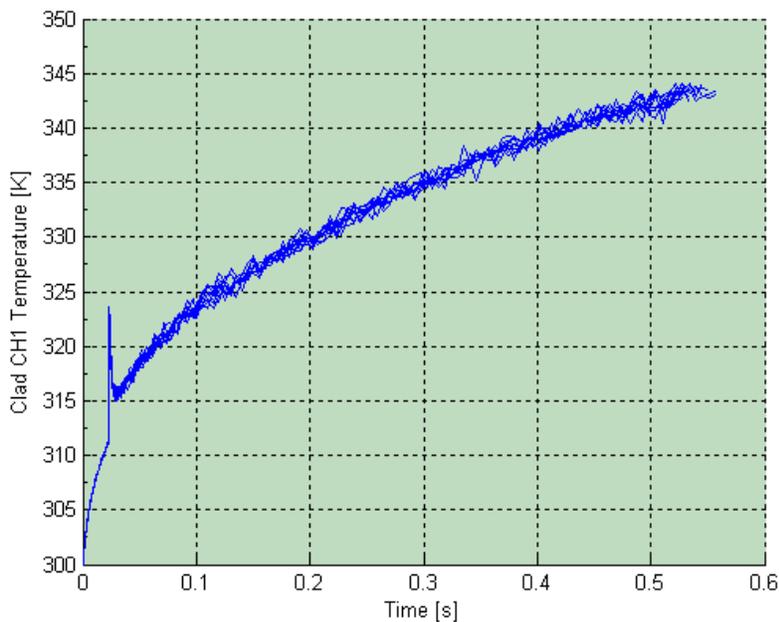


Figure 25: Clad temperature behavior for the hot channel

5 FUTURE DEVELOPMENTS

5.1 General Software Framework

While most of the framework has been already build this year, it is needed to strength the regression test batch and to define an automatized process to mirror the RELAP-7 test batch. During this year it has been highlighted that given the deep interdependence of the two codes and the fast pace of development, it very work intensive to keep test batches in sync especially when changes are made on already existing components. This point is also crucial of one of next year task that will be to define a solid Quality Assurance model for any further development.

5.2 Control Logic

The software infrastructure for the control logic has been already completed this year but of course not the specific implementation for each RELAP-7 component. In this respect, next year will focus on:

- Ensuring that the software interface that allow to access variable that needs to be controlled is in place in all current RELAP-7 components and in the ones will developed during next years
- Building an internal list (enumerator) of 'approved' variable for control. While for uncertainty quantification it might be allowed to perturb all parameters in all equation for control purpose this should be limited to avoid to alter the fundamental models inside the code
- Adding post-processing capabilities, in particular the ones requiring time integration and differentiation. As seen in the example of the PID controller these quantities are fundamental to the application of the control logic

Another important future task will be the creation of a library of functions accessible to the user via the Python interface. Two different type of libraries are envisioned:

- Component library:
 - The component library will contain several commonly used control function for components, it will range from pump and valve characteristics to decay heat curves.
- Statistical function libraries:
 - This library will contain a set of random generators, probability distribution functions, cumulative distribution functions and their inverse.

5.3 Probabilistic Analysis

While the set of equations to be solved has been defined the solution capability presently is via Monte Carlo sampling. It is well know that this methodology is not performing effectively when the number of sampled parameter grows above few tens and moreover statistics might be very low in high risk low probability areas of the phase space. In order to overcome this issue, several activities will be performed next year.

5.3.1 Dynamic event tree implementation

The event trees approach has been already described in the software implementation for the probabilistic analysis and will be put in place in the next fiscal year. In this respect, handling large number of file job queuing and load balance in the parallel implementation will represent the main challenges; at the moment there are not show stoppers foreseen for the implementation of this methodology in the current software implementation.

5.3.2 Data Mining

The large number of simulation that will be needed to deeply investigate the plant behavior with a reasonable statistics will create a large amount of data of which not all will be relevant. Linear and not linear coordinate transformation of the phase space (Principal Component Analysis and Kernel Principal Component analysis) to reduce dimensionality. In order to reduce the cardinality of the sampling also clustering algorithm should be explored and implemented.

Another relevant aspect of data mining is to use the information to trace back driver for high-risk situations and source of uncertainties. To perform such analysis it will be needed to evaluate sensitivity coefficients toward risk functions and statistical dispersion. This task is rather large and therefore will be more a multi year line of development but non the less it should be possible to see preliminary results at the end of next fiscal year.

5.3.3 Guided sampling strategies

An innovative approach to increase the effectiveness of Monte Carlo sampling is via an appropriate sampling strategy. Several algorithms have been proposed in this direction so far in concurrent projects. A common ground of those algorithms is to use the sensitivity coefficient toward risk function to prefer certain region of the initial condition space.

At this moment, it is not possible to identify a specific choice in term of algorithm for this task. Therefore next year the development of data mining techniques will also to better characterize the behavior of the system under consideration and therefore provide an indication of the possible candidate.

5.4 GUI

Provided the above envisioned further developments the GUI plan should be derived according. It is also a given that the tasks below mentioned are not meant to be completed over the next fiscal year but they will span over the entire life time of the project

- RELAP-7 input support
 - Increase the number of components for which graphical visualization is possible to match current and future component availability in RELAP-7
 - Capability to duplicate components
 - Drag and drop of component in the plant layout
- Control Logic
 - Create drop down list of controllable variables
 - Create drop down list of monitored variables
 - Create a drop down list of the auxiliary variables
 - Create a specific window for the implementation of the control logic with available the list of:
 - Statistical functions in the library (R interface)
 - Component control laws in the library
 - Access to the list of auxiliary, monitored, and controlled variables
 - Create a mask for the implementation of the control logic that will make transparent to the user the Python scripting (direct Python scripting still available)
- Probabilistic analysis
 - Full support for multi-platform parallel Monte-Carlo sampling
 - Input for branching criteria used in tree analysis
 - Enhance visualization to allow for multi runs comparison
 - Add a specific windows to perform data post processing (data mining)
- Solution visualization and data post processing
 - Extend the capability of the 'visualize' windows to allow for data manipulation (possibly interface with Paraview)

A table summarizing needed effort by function and operational area is summarized in appendix B

6 Set Up PRA demo

While it is not within the scope of this project to perform detailed PRA analysis of nuclear plant given that this will be one of the most relevant final application of the code it is meaning full to test the project against a problem representative of this class of user needs.

The demo is build on a simplify PWR plan scheme (Figure 26). The plant simulation starts from nominal power condition but in not equilibrium thermodynamically. The steady state reaching is determined on the stabilization of the clad temperature (Figure 27) that is one of the most sensible values to oscillation in the coolant. Once the steady state is reached (~50 seconds) the plant status is saved and used as a re-start fro the PRA analysis.

The transient to be analyzed by the PRA is the following

- Initial loss of external power at 51 sec
- Reactor scram and decay heat curve are used to determine the residual power decay
- Primary pump coast down start (pumps slow down until 0.1% of their nominal head than head is set to zero)

- Assuming that diesel will not start after 3 sec the probability of later start is given by a Gaussian centered in 5 minutes with 10% sigma
- The simulation ends if the clad temperature exceeds a temperature determined by a Gaussian distribution with mean value provided by the user (this is chosen as a degree of freedom to be determined to avoid boiling due to the lack of two phase flow)
- The simulation ends 10 minutes after the loss of power

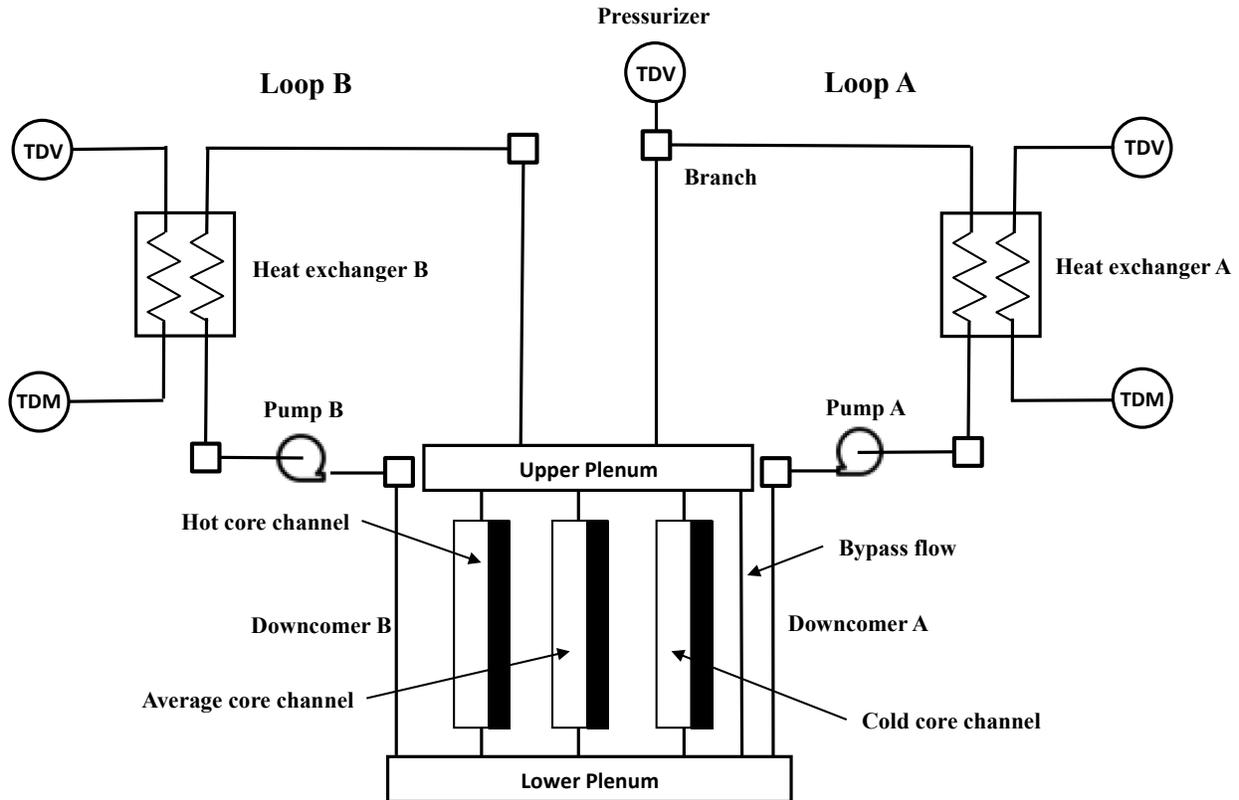


Figure 26: PWR reference scheme

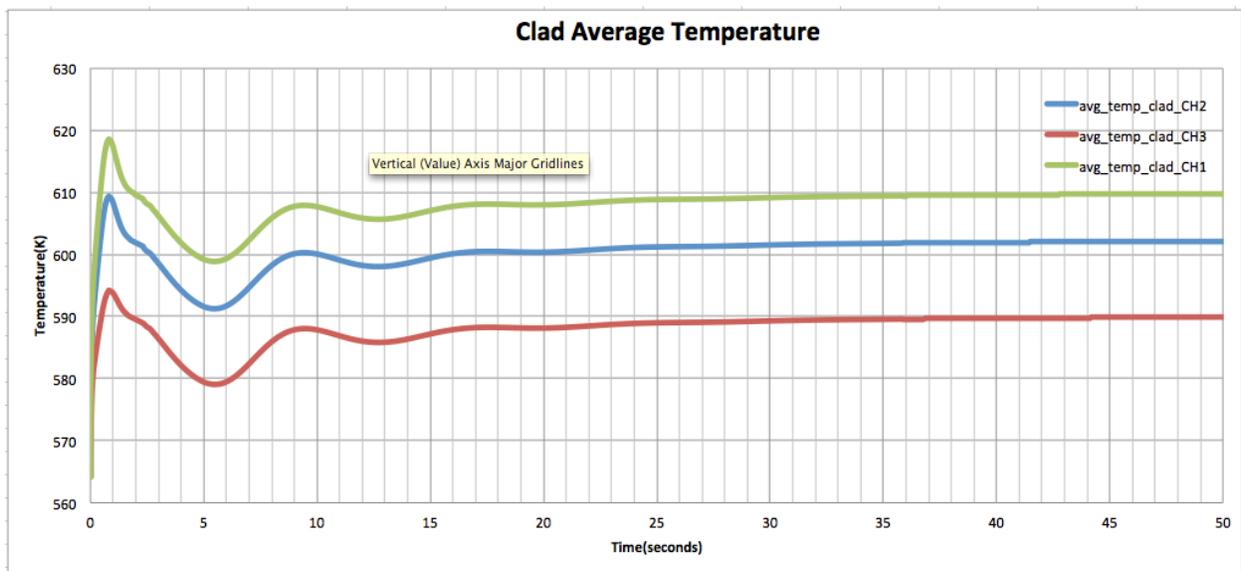


Figure 27: Time evolution of clad temperature till steady state is reached

7 APPENDIX A: RELAP-7 RAVEN Input File and Control Logic:

In this appendix it will be shown the RELAP 7 input file for a demo based on a general PWR reactor as also the corresponding control logic to perform uncertainty and PRA analysis.

RELAP 7 RAVEN Input

```
[GlobalParams]
model_type = 3
global_init_P = 15.17e6
global_init_V = 0.
global_init_T = 564.15
scaling_factor_var = '1.e-1 1.e-5 1.e-8'
initial_RP = 2.77199979e9
[]

[EoS]
[./eos]
type = NonIsothermalEquationOfState
p_0 = 15.17e6 # Pa
rho_0 = 738.350 # kg/m^3
a2 = 1.e7 # m^2/s^2
beta = .46e-3 # K^{-1}
cv = 5.832e3 # J/kg-K
e_0 = 3290122.80 # J/kg
T_0 = 564.15 # K
[../]
[]

[Components]
[./CH1]
type = CoreChannel
eos = eos
position = '0 -1.2 0'
orientation = '0 0 1'
A = 1.161864
Dh = 0.01332254
length = 3.6576
n_elems = 8
f = 0.01
Hw = 5.33e4
aw = 276.5737513
Ts_init = 564.15
n_heatstruct = 3
name_of_hs = 'FUEL GAP CLAD'
fuel_type = cylinder
width_of_hs = '0.0046955 0.0000955 0.000673'
elem_number_of_hs = '3 1 1'
k_hs = '3.65 1.084498 16.48672'
Cp_hs = '288.734 1.0 321.384'
rho_hs = '1.0412e2 1.0 6.6e1'
power_fraction = '3.33672612e-1 0 0'
[../]
[./CH2]
type = CoreChannel
eos = eos
position = '0 0 0'
orientation = '0 0 1'
A = 1.549152542
Dh = 0.01332254
length = 3.6576
n_elems = 8
f = 0.01
Hw = 5.33e4
aw = 276.5737513
Ts_init = 564.15
n_heatstruct = 3
name_of_hs = 'FUEL GAP CLAD'
fuel_type = cylinder
width_of_hs = '0.0046955 0.0000955 0.000673'
elem_number_of_hs = '3 1 1'
k_hs = '3.65 1.084498 16.48672'
Cp_hs = '288.734 1.0 321.384'
rho_hs = '1.0412e2 1.0 6.6e1'
power_fraction = '3.69921461e-1 0 0'
[../]
[./CH3]
type = CoreChannel
eos = eos
position = '0 1.2 0'
orientation = '0 0 1'
A = 1.858983051
Dh = 0.01332254
length = 3.6576
n_elems = 8
f = 0.01
Hw = 5.33e4
aw = 276.5737513
Ts_init = 564.15
n_heatstruct = 3
name_of_hs = 'FUEL GAP CLAD'
fuel_type = cylinder
width_of_hs = '0.0046955 0.0000955 0.000673'
elem_number_of_hs = '3 1 1'
k_hs = '3.65 1.084498 16.48672'
Cp_hs = '288.734 1.0 6.6e3'
rho_hs = '1.0412e2 1.0 6.6e1'
power_fraction = '2.96405926e-1 0 0'
[../]
[./bypass_pipe]
type = Pipe
eos = eos
position = '0 1.5 0'
orientation = '0 0 1'
A = 1.589571014
Dh = 1.42264
length = 3.6576
n_elems = 5
f = 0.001
Hw = 0.0
[../]
[./LowerPlenum]
type = ErgBranch
eos = eos
inputs = 'DownComer-A(out) DownComer-B(out)'
outputs = 'CH1(in) CH2(in) CH3(in) bypass_pipe(in)'
K = '0.2 0.2 0.2 0.2 0.4 40.0'
Area = 3.618573408
Initial_pressure = 151.7e5
[../]
[./UpperPlenum]
type = ErgBranch
eos = eos
inputs = 'CH1(out) CH2(out) CH3(out) bypass_pipe(out)'
outputs = 'pipe1-HL-A(in) pipe1-HL-B(in)'
K = '0.5 0.5 0.5 80.0 0.5 0.5'
Area = 7.562307456
Initial_pressure = 151.7e5
[../]
[./DownComer-A]
type = Pipe
```

```

eos = eos
position = '0 2.0 4.0'
orientation = '0 0 -1'
A = 3.6185734
Dh = 1.74724302
length = 4
n_elems = 3
f = 0.001
Hw = 0.
[../]
[./pipe1-HL-A]
type = Pipe
eos = eos
position = '0 0.5 4.0'
orientation = '0 0 1'
A = 7.562307456
Dh = 3.103003207
length = 4.
n_elems = 3
f = 0.001
Hw = 0.0
[../]
[./pipe2-HL-A]
type = Pipe
eos = eos
position = '0 0.5 8.0'
orientation = '0 1 0'
A = 2.624474
Dh = 1.828
length = 3.5
n_elems = 3
f = 0.001
Hw = 0.0
[../]
[./pipe1-CL-A]
type = Pipe
eos = eos
position = '0 3.0 4.0'
orientation = '0 -1 0'
A = 2.624474
Dh = 1.828
length = 1.
n_elems = 3
f = 0.001
Hw = 0.0
[../]
[./pipe2-CL-A]
type = Pipe
eos = eos
position = '0 4 4.0'
orientation = '0 -1 0'
A = 2.624474
Dh = 1.828
length = 0.8
n_elems = 3
f = 0.001
Hw = 0.0
[../]
[./pipe1-SC-A]
type = Pipe
eos = eos
position = '0 5.2 4.0'
orientation = '0 -1 0'
A = 2.624474
Dh = 1.828
length = 1.
n_elems = 3
f = 0.001
Hw = 0.0
[../]
[./pipe2-SC-A]
type = Pipe

```

```

eos = eos
position = '0 4.2 8.0'
orientation = '0 1 0'
A = 2.624474
Dh = 1.828
length = 1.
n_elems = 3
f = 0.001
Hw = 0.0
[../]
[./Branch1-A]
type = ErgBranch
eos = eos
inputs = 'pipe1-HL-A(out)'
outputs = 'pipe2-HL-A(in) pipe-to-Pressurizer(in)'
K = '0.5 0.7 80.'
Area = 7.562307456
Initial_pressure = 151.7e5
[../]
[./Branch2-A]
type = ErgBranch
eos = eos
inputs = 'pipe1-CL-A(out)'
outputs = 'DownComer-A(in)'
K = '0.5 0.7'
Area = 3.6185734
Initial_pressure = 151.7e5
[../]
[./Branch3-A]
type = ErgBranch
eos = eos
inputs = 'pipe2-HL-A(out)'
outputs = 'HX-A(primary_in)'
K = '0.5 0.7'
Area = 2.624474
Initial_pressure = 151.7e5
[../]
[./Pump-A]
type = Pump
eos = eos
Area = 2.624474
Initial_pressure = 151.7e5
Head = 9.9
K_reverse = 1000
outlet = 'pipe1-CL-A(in)'
inlet = 'pipe2-CL-A(out)'
[../]
[./HX-A]
type = HeatExchanger
eos = eos
eos_secondary = eos
position = '0 4. 8.'
orientation = '0 0 -1'
A = 5.
A_secondary = 5.
Dh = 0.01
Dh_secondary = 0.01
length = 4.
n_elems = 10
Hw = 1.e4
Hw_secondary = 1.e4
aw = 539.02
aw_secondary = 539.02
f = 0.01
f_secondary = 0.01
Twall_init = 564.15
wall_thickness = 0.001
k_wall = 100.0
rho_wall = 100.0
Cp_wall = 100.0
n_wall_elems = 2
[../]

```

```

[./Branch4-A]
type = ErgBranch
eos = eos
inputs = 'pipe1-SC-A(out)'
outputs = 'HX-A(secondary_in)'
K = '0.5 0.7'
Area = 2.624474e2
Initial_pressure = 151.7e5
[../]
[./Branch5-A]
type = ErgBranch
eos = eos
inputs = 'HX-A(secondary_out)'
outputs = 'pipe2-SC-A(in)'
K = '0.5 0.7'
Area = 2.624474e2
Initial_pressure = 151.7e5
[../]
[./Branch6-A]
type = ErgBranch
eos = eos
inputs = 'HX-A(primary_out)'
outputs = 'pipe2-CL-A(in)'
K = '0.5 0.7'
Area = 2.624474e2
Initial_pressure = 151.7e5
[../]
[./PressureOutlet-SC-A]
type = TimeDependentVolume
input = 'pipe2-SC-A(out)'
p_bc = '151.7e5'
T_bc = 564.15
eos = eos
[../]
[./DownComer-B]
type = Pipe
eos = eos
position = '0 -2.0 4.0'
orientation = '0 0 -1'
A = 3.6185734
Dh = 1.74724302
length = 4
n_elems = 3
f = 0.001
Hw = 0.
[../]
[./pipe1-HL-B]
type = Pipe
eos = eos
position = '0 -0.5 4.0'
orientation = '0 0 1'
A = 7.562307456
Dh = 3.103003207
length = 4.
n_elems = 3
f = 0.001
Hw = 0.0
[../]
[./pipe2-HL-B]
type = Pipe
eos = eos
position = '0 -0.5 8.0'
orientation = '0 -1 0'
A = 2.624474
Dh = 1.828
length = 3.5
n_elems = 3
f = 0.001
Hw = 0.0
[../]
[./pipe1-CL-B]
type = Pipe
eos = eos
position = '0 -3.0 4.0'
orientation = '0 1 0'
A = 2.624474
Dh = 1.828
length = 1.
n_elems = 3
f = 0.001
Hw = 0.0
[../]
[./pipe2-CL-B]
type = Pipe
eos = eos
position = '0 -4.0 4.0'
orientation = '0 1 0'
A = 2.624474
Dh = 1.828
length = 0.8
n_elems = 3
f = 0.001
Hw = 0.0
[../]
[./pipe1-SC-B]
type = Pipe
eos = eos
position = '0 -5.2 4.0'
orientation = '0 1 0'
A = 2.624474
Dh = 1.828
length = 1.
n_elems = 3
f = 0.001
Hw = 0.0
[../]
[./pipe2-SC-B]
type = Pipe
eos = eos
position = '0 -4.2 8.0'
orientation = '0 -1 0'
A = 2.624474
Dh = 1.828
length = 1.
n_elems = 3
f = 0.001
Hw = 0.0
[../]
[./Branch1-B]
type = ErgBranch
eos = eos
inputs = 'pipe1-HL-B(out)'
outputs = 'pipe2-HL-B(in)'
K = '0.5 0.7'
Area = 7.562307456
Initial_pressure = 151.7e5
[../]
[./Branch2-B]
type = ErgBranch
eos = eos
inputs = 'pipe1-CL-B(out)'
outputs = 'DownComer-B(in)'
K = '0.5 0.7'
Area = 3.6185734
Initial_pressure = 151.7e5
[../]
[./Branch3-B]
type = ErgBranch
eos = eos
inputs = 'pipe2-HL-B(out)'
outputs = 'HX-B(primary_in)'
K = '0.5 0.7'
Area = 2.624474
Initial_pressure = 151.7e5

```

```

[../]
[./Pump-B]
type = Pump
eos = eos
Area = 2.624474
Initial_pressure = 151.7e5
Head = 9.9
K_reverse = 1000
outlet = 'pipe1-CL-B(in)'
inlet = 'pipe2-CL-B(out)'
[../]
[./HX-B]
type = HeatExchanger
eos = eos
eos_secondary = eos
position = '0 -4. 8.'
orientation = '0 0 -1'
A = 5.
A_secondary = 5.
Dh = 0.01
Dh_secondary = 0.01
length = 4.
n_elems = 10
Hw = 1.e4
Hw_secondary = 1.e4
aw = 539.02
aw_secondary = 539.02
f = 0.01
f_secondary = 0.01
Twall_init = 564.15
wall_thickness = 0.001
k_wall = 100.0
rho_wall = 100.0
Cp_wall = 100.0
n_wall_elems = 2
disp_mode = -1.0
[../]
[./Branch4-B]
type = ErgBranch
eos = eos
inputs = 'pipe1-SC-B(out)'
outputs = 'HX-B(secondary_in)'
K = '0.5 0.7'
Area = 2.624474e2
Initial_pressure = 151.7e5
[../]
[./Branch5-B]
type = ErgBranch
eos = eos
inputs = 'HX-B(secondary_out)'
outputs = 'pipe2-SC-B(in)'
K = '0.5 0.7'
Area = 2.624474e2
Initial_pressure = 151.7e5
[../]
[./Branch6-B]
type = ErgBranch
eos = eos
inputs = 'HX-B(primary_out)'
outputs = 'pipe2-CL-B(in)'
K = '0.5 0.7'
Area = 2.624474e2
Initial_pressure = 151.7e5
[../]
[./PressureOutlet-SC-B]
type = TimeDependentVolume
input = 'pipe2-SC-B(out)'
p_bc = '151.7e5'
T_bc = 564.15
eos = eos
[../]
[./pipe-to-Pressurizer]
type = Pipe
eos = eos
position = '0 0.5 8.0'
orientation = '0 0 1'
A = 2.624474
Dh = 1.828
length = 0.5
n_elems = 3
f = 10.
Hw = 0.0
[../]
[./Pressurizer]
type = TimeDependentVolume
input = 'pipe-to-Pressurizer(out)'
p_bc = '151.7e5'
T_bc = 564.15
eos = eos
[../]
[./high_pressure_secondary_A]
T_bc = 537.15
p_bc = '152.19e5'
eos = eos
input = 'pipe1-SC-A(in)'
type = TimeDependentVolume
[../]
[./high_pressure_secondary_B]
T_bc = 537.15
p_bc = '152.19e5'
eos = eos
input = 'pipe1-SC-B(in)'
type = TimeDependentVolume
[../]
[]
[Preconditioning]
active = 'FDP_PJFNK'
[./FDP_PJFNK]
type = FDP
full = true
petsc_options = '-snes_mf_operator -pc_factor_shift_nonzero'
petsc_options_iname = '-mat_fd_type'
petsc_options_value = 'ds'
petsc_options_iname = '-mat_fd_type'
petsc_options_value = 'ds'
[../]
[./FDP_Newton]
type = FDP
full = true
petsc_options = '-snes'
petsc_options_iname = '-mat_fd_type'
petsc_options_value = 'ds'
petsc_options_iname = '-mat_fd_type'
petsc_options_value = 'ds'
[../]
[]
[Executioner]
type = RavenExecutioner
dt = 5e-2
time_t = '0 3.0 5.01 9.5 9.75 14 17 5e1'
time_dt = '1.e-2 1.0 1.1 2.5e-1 2.5e-1 2.5e-1 2.5e-1 2.5e-1'
dtmax = 9999
e_tol = 10.0
e_max = 99999.
max_increase = 2
perf_log = true
petsc_options_iname = '-ksp_gmres_restart'
petsc_options_value = '300' # '300'
nl_rel_tol = 1e-6
nl_abs_tol = 1e-10
nl_max_its = 100
l_tol = 1e-5 # Relative linear tolerance for each Krylov solve

```

```

l_max_its = 100 # Number of linear iterations for each Krylov solve
start_time = 0.0
end_time = 50
ss_check_tol = 1e-05
nl_rel_step_tol = 1e-3
[/Quadrature]
type = TRAP
order = FIRST
[../]
[]

```

```

[Output]
file_base = TMI_test_PRA_steady_state
exodus = true
output_initial = true
perf_log = true
num_restart_files = 1
[]

```

```

[Controlled]
control_logic_input = TMI_test_PRA_control
[/power_CH1]
print_csv = true
property_name = total_power_scaling
data_type = double
component_name = CH1
[../]
[/power_CH2]
property_name = total_power_scaling
print_csv = true
data_type = double
component_name = CH2
[../]
[/power_CH3]
property_name = total_power_scaling
print_csv = true
data_type = double
component_name = CH3
[../]
[/high_pressure_secondary_A]
property_name = p_in
print_csv = true
data_type = double
component_name = high_pressure_secondary_A
[../]
[/high_pressure_secondary_B]
property_name = p_in
print_csv = true
data_type = double
component_name = high_pressure_secondary_B
[../]
[/Head_PumpB]
property_name = Head
data_type = double
print_csv = true
component_name = Pump-B
[../]
[/Head_PumpA]
property_name = Head
data_type = double
print_csv = true
component_name = Pump-A
[../]
[]

```

```

[Monitored]
[/max_temp_clad_CH1]
operator = NodalMaxValue
path = CLAD:TEMPERATURE
data_type = double
component_name = CH1
[../]

```

```

[/max_temp_clad_CH2]
operator = NodalMaxValue
path = CLAD:TEMPERATURE
data_type = double
component_name = CH2
[../]
[/max_temp_clad_CH3]
operator = NodalMaxValue
path = CLAD:TEMPERATURE
data_type = double
component_name = CH3
[../]
[/Max_Fluid_Vel_H_L-A]
operator = NodalMaxValue
path = VELOCITY
data_type = double
component_name = pipe1-HL-A
[../]
[/Max_Fluid_Vel_C_L_A]
operator = NodalMaxValue
path = VELOCITY
data_type = double
component_name = DownComer-A
[../]
[/out_temp_sec_A]
operator = ElementAverageValue
path = TEMPERATURE
data_type = double
component_name = pipe2-SC-A
[../]
[/DownStreamSpeed]
operator = ElementAverageValue
path = VELOCITY
data_type = double
component_name = pipe1-CL-B
[../]
[/UpstreamSpeed]
operator = ElementAverageValue
path = VELOCITY
data_type = double
component_name = pipe1-CL-B
[../]
[/max_temp_fuel_CH1]
operator = NodalMaxValue
path = FUEL:TEMPERATURE
data_type = double
component_name = CH1
[../]
[/max_temp_fuel_CH2]
operator = NodalMaxValue
path = FUEL:TEMPERATURE
data_type = double
component_name = CH2
[../]
[/max_temp_fuel_CH3]
operator = NodalMaxValue
path = FUEL:TEMPERATURE
data_type = double
component_name = CH3
[../]
[]

```

```

[PredefinedDistributions]
[/trip_dist]
parameter1 = 1
parameter2 = 0.1
x_max = 2
x_min = 0
type = NORMAL
[../]
[/1%_gauss]
parameter1 = 1

```

```

parameter2 = 0.01
x_max = 0.8
x_min = 0
type = NORMAL
[../]
[]

[RavenAuxiliary]
[./scram_start_time]
data_type = double
initial_value = 10000
print_csv = true
[../]
[./auxiliary_system_time_on]
data_type = double
initial_value = 5
print_csv = true
[../]
[./InitialMassFlowPrimary]
data_type = double
initial_value = 0
print_csv = true
[../]
[./initialInletSecPress]
data_type = double
print_csv = true
initial_value = 0
[../]
[./CladDamaged]
data_type = bool
print_csv = true
initial_value = False
[../]
[./AuxDieselAvailable]
data_type = bool
print_csv = true
initial_value = true
[../]
[./AuxDieselSupply]
data_type = bool
print_csv = true
initial_value = true
[../]
[./DeltaTimeScramToAux]
data_type = double
initial_value = 4
print_csv = true
[../]

[./InitialOutletSecPress]
data_type = double
print_csv = true
initial_value = 0
[../]
[./AuxDieselRecoveryTime]
data_type = double
print_csv = true
initial_value = 180
[../]
[./PrimaryPump]
data_type = double
print_csv = true
initial_value = 0
[../]
[./PrimaryPumpTransStart]
data_type = double
print_csv = true
initial_value = 0.5
[../]
[./SecondaryPump]
data_type = double
print_csv = true
initial_value = 0
[../]
[./SecondaryPumpTransStart]
data_type = double
print_csv = true
initial_value = 0.5
[../]
[./PowerStatus]
data_type = double
print_csv = true
initial_value = 0
[../]
[./CladTempTreshold]
data_type = double
print_csv = true
initial_value = 1477.59
[../]
[./HydroPressureSecondary]
data_type = double
print_csv = true
initial_value = 29.35152e+3
[../]
[]

```

7.1 Control Logic Python File

```

import sys
import math
import distribution1D
import raventools
distcont = distribution1D.DistributionContainer.Instance()
DecayHeatScalingFactor = raventools.decayHeat(1,1,3600*24*30*8,0.064)
PumpCoastDown = raventools.pumpCoastdown(1,9.9)
PumpCoastDown Sec = raventools.pumpCoastdown(1,1)

def initial_function(monitored, controlled, auxiliary):

#Nominal/initial values
auxiliary.InitialHead = controlled.Head_PumpB
auxiliary.initialInletSecPress = controlled.high_pressure_secondary_A
return

def control_function(monitored, controlled, auxiliary):

if monitored.time>=auxiliary.scram_start_time:

```

```

auxiliary.ScramStatus = True
else:
    auxiliary.ScramStatus = False

if auxiliary.ScramStatus: #we are in scram situation
    #primary pump B
    if controlled.Head_PumpB>1.e-4*auxiliary.InitialHead:
        controlled.Head_PumpB = PumpCoastDown.flowrateCalculation(monitored.time-auxiliary.scram_start_time)
    else:
        controlled.Head_PumpB = 0
    #primary pump A
    if controlled.Head_PumpA>1.e-4*auxiliary.InitialHead:
        controlled.Head_PumpA = PumpCoastDown.flowrateCalculation(monitored.time-auxiliary.scram_start_time)
    else:
        controlled.Head_PumpA = 0
    #core power following decay heat curve
    controlled.power_CH1 =DecayHeatScalingFactor.powerCalculation(monitored.time-auxiliary.scram_start_time)
    controlled.power_CH2 =DecayHeatScalingFactor.powerCalculation(monitored.time-auxiliary.scram_start_time)
    controlled.power_CH3 =DecayHeatScalingFactor.powerCalculation(monitored.time-auxiliary.scram_start_time)
    #secondary system replaced by auxiliary secondary system
    if monitored.time<(auxiliary.scram_start_time+auxiliary.DeltaTimeScramToAux): # not yet auxiliary system up
        if controlled.high_pressure_secondary_A >= auxiliary.InitialOutletSecPress:
            if auxiliary.initialInletSecPress*PumpCoastDownSec.flowrateCalculation(monitored.time-auxiliary.scram_start_time) >=
                (auxiliary.InitialOutletSecPress+ auxiliary.HydroPressureSecondary):
                controlled.high_pressure_secondary_A = auxiliary.initialInletSecPress*
                    PumpCoastDownSec.flowrateCalculation(monitored.time-auxiliary.scram_start_time)
            else:
                controlled.high_pressure_secondary_A = auxiliary.InitialOutletSecPress+ auxiliary.HydroPressureSecondary
        else:
            controlled.high_pressure_secondary_A = auxiliary.InitialOutletSecPress+ auxiliary.HydroPressureSecondary
        if controlled.high_pressure_secondary_B >= auxiliary.InitialOutletSecPress:
            if auxiliary.initialInletSecPress*PumpCoastDownSec.flowrateCalculation(monitored.time-auxiliary.scram_start_time) >=
                (auxiliary.InitialOutletSecPress+ auxiliary.HydroPressureSecondary):
                controlled.high_pressure_secondary_B = auxiliary.initialInletSecPress*
                    PumpCoastDownSec.flowrateCalculation(monitored.time-auxiliary.scram_start_time)
            else:
                controlled.high_pressure_secondary_B = auxiliary.InitialOutletSecPress+ auxiliary.HydroPressureSecondary
        else:
            controlled.high_pressure_secondary_B = auxiliary.InitialOutletSecPress+ auxiliary.HydroPressureSecondary
    else: # auxiliary system up
        if controlled.high_pressure_secondary_A<15198299.45: #check if it has not already reached the final value
            controlled.high_pressure_secondary_A = auxiliary.InitialOutletSecPress +
                (15198299.45-auxiliary.InitialOutletSecPress)/5*
                (monitored.time-(auxiliary.scram_start_time+auxiliary.DeltaTimeScramToAux))
        if controlled.high_pressure_secondary_B<15198299.45: #check if it has not already reached the final value
            controlled.high_pressure_secondary_B = auxiliary.InitialOutletSecPress +
                (15198299.45-auxiliary.InitialOutletSecPress)/5*
                (monitored.time-(auxiliary.scram_start_time+auxiliary.DeltaTimeScramToAux))
    else:
        return
if (monitored.max_temp_clad_CH1>auxiliary.CladTempTreshold) or
(monitored.max_temp_clad_CH2>auxiliary.CladTempTreshold) or
(monitored.max_temp_clad_CH3>auxiliary.CladTempTreshold):
    auxiliary.CladDamaged = True
    raise NameError ('exit condition reached - failure of the clad')
return

```

8 APPENDIX B

The following table summarizes the needs and the respective area of development interested that are deemed necessary to reach a reasonable level interest from possible users.

Work Area	Quality Assurance	RELAP 7 Input GUI	RAVEN Control Logic	Data Mining	PRA
Software Framework	-Linking Regression Test to RELAP 7 -Installation/Release package	DONE	-Construct templates for control functions -Registration of control functions as MOOSE actions (?) -Allow stop/start with recompiled control logic	-R evaluation/ integration (interface) - Construct templates for data processing functions -Template registration of statistical functions as MOOSE actions (?)	-Construct templates for statistical functions - Template registration of statistical functions as MOOSE actions (?) -R evaluation/ integration -Automatic simulation branching for tree constructions
Software implementation	-Verify doxygen implementation -build regression test to achieve >90% coverage	-Implementation of enumerators for all controllable variable -Drop down menus for the definition of controllable, and monitorable variables	-Extension of MOOSE post processors (integral and differential over time, coordinate return) -Building a library of control laws for components -Registration of each component control function (?)	-Building library of data mining functions -Registration of each data mining function (?)	-Construction a library of statistical functions (in case of R integration building interfaces)
Theory development			-Signal analysis for simulation of plant operation control	-Evaluate effectiveness of different data mining approaches	-Automatic branching strategies -Adaptive goal oriented sampling -Adaptive goal oriented branching
GUI implementation		-Extend graphical coverage to missed component and future ones -Create graphical interactive operators (duplicate, drag and drop) -Interface visualization with Paraview for post processing solution data (?)	-Build GUI for control logic input and Python automatic generation -Construct interface for RELAP stop/start control and control logic re-compilation -online signal analysis capability	-Create a new tabs for data mining -Allow uploading of simulations data bases	-Visualization of data while parallel simulation are ongoing -Visualization of data mining results while data base is building up
Visible Outcome	-Manuals -QA documents	-Full graphical input creation -Data post processing interface	-Full graphical input of the control logic -Control logic function library -Visualization of results of signal analysis	-Capability to perform data mining on simulation data bases via GUI	-In line visualization of simultaneous simulation results and inline data mining -Adaptive PRA capability

(?) Task with a question mark implies that a final decision still needs to be taken on their implementation